Universidad Politécnica de Madrid

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

POLITÉCNICA

Universidad Politécnica de Madrid

ETSI SISTEMAS INFORMÁTICOS

Grado en Ingeniería de Computadores

Sistema de Seta de Emergencia Integrada en Vehículos

Proyecto Fin de Grado

Autor:
Rubén Heras García

Director:
Alfredo Valle Barrio

Tutor:
Juan Arquero Gallego

Curso 2023-2024

Agradecimientos

Doy las gracias a Juan Arquero Gallego mi cotutor, que ha hecho posible la integración del proyecto en la PCB, ayudándome tanto en la elección de los componentes como en el diseño, en mis preguntas sobre la memoria y dándome consejos. A Óscar González Fuentes con el que he hecho posible el correcto funcionamiento de la seta, ayudándome a soldar todos los componentes necesarios, haciendo pruebas hasta cansarnos y enseñándome cómo se hace un buen trabajo. A Guillermo Sánchez Gutiérrez por darme consejos sobre el software y cómo hacer una buena memoria. A Alfredo Valle por hacer una muy buena corrección de lo que se muestra en este documento y por confiar en mí para realizar proyectos. A todas las personas que me han apoyado durante este proyecto y durante mi carrera, mis amigos y mis familiares.

Especial agradecimiento al INSIA el cual ha financiado y ofrecido herramientas para este proyecto y me ha dado una oportunidad para mostrar mis conocimientos en un mundo laboral y profesional.

Resumen

En este proyecto se elaborará un sistema de setas de emergencia que al ser pulsadas enviarán información a la centralita de un vehículo autónomo a través del bus CAN para realizar diferentes funciones como la activación del frenado de emergencia o la desconexión del control autónomo del vehículo.

El objetivo es incrementar la seguridad en los campos de pruebas de vehículos autónomos y la mejora de la seguridad vial en vehículos cotidianos, en el que se propone desarrollar un sistema innovador de detección y prevención de situaciones de emergencia, con el fin de reducir la incidencia de accidentes y mejorar la respuesta ante eventos imprevistos en el ámbito automotriz.

El proyecto ha tenido como motivación la creación de un sistema hardware que responda a situaciones en tiempo real el cual permita actuar sobre el vehículo con el objetivo de mejorar la respuesta ante situaciones imprevistas mejorando de esta forma la seguridad vial.

El sistema se ha diseñado e impreso en PCB y montado en una estructura robusta para obtener un acabado final profesional y compacto. La seta de emergencia incorpora dos switches para el armado y desarmado del sistema, un DipSwitch para la selección del número de seta, un buzzer para un aviso sonoro cuando el sistema está pulsado y un envío de mensajes CAN según el estado del sistema a través de la placa de desarrollo MCP2515. Todo esto controlado por un Arduino Nano que ha sido integrado y soldado en la placa.

En esta memoria se expondrán los pasos realizados para el desarrollo del sistema comenzando con la historia de la seguridad desde que apareció el concepto del vehículo, repasando los sistemas de seguridad actuales y los protocolos de comunicación que incorporan hoy en día los vehículos. Seguidamente se desarrollará el proceso de elaboración del sistema detallando cómo se han diseñado los circuitos eléctricos para la integración del Hardware, la programación del código a lo que se refiere el Software y por último los resultados y conclusiones que se han obtenido a lo largo del desarrollo.

En conclusión, durante este proyecto se ha elaborado un sistema de setas de emergencia para vehículos autónomos que representa un avance en la mejora de la seguridad vial. A través del desarrollo Hardware y Software, el sistema ofrece una solución efectiva para detectar y prevenir situaciones de peligro, contribuyendo así a la reducción de accidentes y la protección de los usuarios de la carretera.

Palabras clave: PCB, DipSwitch, CAN, MCP2515, Arduino Nano

Abstract

In this project, an emergency button system will be developed, which when pressed will send information to the autonomous vehicle´s control unit via the CAN bus to perform various functions such as activating emergency braking or disconnecting the vehicle´s autonomous control.

The aim to increase safety in autonomous vehicles testing fields and improve road safety in street vehicles, proposing to develop an innovative system for detecting and preventing emergency situations, with de goal of reducing accident rates and improving responses to unforeseen events in the automotive field.

The project has been motivated by the creation of a hardware system that responds to real-time situations, allowing for immediate action on the vehicle with the aim of improving the response to unforeseen events, thereby enhancing road safety.

The system has been designed, printed on PCB, and assembled on a robust structure to achieve a professional and compact final product. The emergency button incorporates two switches for arming and disarming de system, a DipSwitch for selecting the button number, a buzzer for audible alerts when the system is activated, and CAN message transmission according to the system´s status trough the MCP2515 development board. All of this controlled by an integrated and soldered Arduino Nano.

This report will outline the steps taken for system development, starting with the history of vehicle safety, reviewing current safety systems, and communication protocols used in today´s vehicles. The development process will then be detailed, covering the design of electrical circuits for Hardware integration, Software code programming, and concluding with the results and findings obtained throughout the development process.

In conclusion, this project has developed an emergency button system for autonomous vehicles, representing an advancement in road safety. Through hardware and software development, the system offers an effective solution for detecting and preventing danger, thereby contributing to accident reduction, and ensuring the safety of road users.

Keywords: PCB, DipSwitch, CAN, MCP2515, Arduino Nano

Índice

Contenido

Índice de Figuras

Índice de Tablas

Acrónimos

NHTSA National Highway Traffic Safety Administration, Administración Nacional de Seguridad del Tráfico en Carreteras.

INSIA Instituto Universitario de Investigación del Automóvil.

CAN Controller Area Network, Red de Área de Control.

ECU Engine Control Unit, Unidad de Control de Motor.

CSMA/CD Carrier Sense Multiple Access with Collision Detection, Acceso Múltiple por Detección de Portadora con Detección de Colisiones.

OSI Open System Interconnection, Modelo de Interconexión de Sistemas Abiertos.

LLC Logical Link Control, Control de Enlace Lógico.

PDO Process Data Objects, Objetos de datos de proceso.

SDO Service Data Objects, Objetos de datos de servicio.

E/S entrada / salida.

NMT Network Management, Gestión de Red.

CiA CAN in Automation, CAN en Automatización.

MOST Media Oriented Systems Transport, Transporte de Sistemas Orientados a Media.

OBD On board Diagnostics , Diagnóstico de abordo.

EOBD European On Board Diagnostics, Diagnóstico de a Bordo Europeo.

SAE Society of Automotive Engineers, Sociedad de Ingenieros de Automoción.

IoT Internet of Things, Internet de las Cosas.

V2X Vehicle to Everything, Vehículo a Todo.

V2I Vehicle to Infraestructure, Vehículo a Infraestructura.

V2V Vehicle to Vehicle, Vehículo a Vehículo.

DSRC Dedicated short-range communications, Comunicación Dedicada de Corto Alcance.

ADAS Advanced Driver Assitance Systems, Sistemas Avanzados de Asistencia al Conductor.

ESP Electronic Stability Programme, Programa Electrónico de Estabilidad.

DDR Dowsiness and distraction Reminder. Sistema de Advertencia de Somnolencia y Distracción.

ISA Intelligence Speed Assitance, Asistente de velocidad Inteligente.

RCTA Rear Cross Traffic Alert, Alerta de tráfico Cruzado.

EDR Event Data Recorder, Grabadora de Datos de Eventos.

LDW Lane departure warning, Alerta de cambio involuntario de carril.

ESS Emergency Signal System,, Sistema de frenado de Emergencia.

GPS Global Positioning System, Sistema de Posicionamiento Global.

BAS Brake Assistant System, Sistema de asistencia de Frenado.

EBA Emergency Brake Assist Asistente de Frenado de Emergencia.

ACC Adaptative Cruise Control, Control de Crucero Adaptativo.

SE Seta de Emergencia.

VSCode Virtual Studio Code, Estudio de Código Virtual.

CLI Command Line Interface, interfaz de línea de comando.

EDA Electronic Design Automation, Automatización de diseño electrónico.

PCB Printed Circuit Board, Tarjeta de Circuito Impreso

VDD Drain to Drain Voltage, Voltaje de drenador a drenador

SPI Serial Peripherial Interface, Serie de Interfaz Periférica.

NEMA National Electrical Manufacturers Association, Asociacón Nacional de Manufacturadores Electrónicos.

SysML Systems Modeling Lenguaje, Lenguaje de Modelado de Sistemas.

UML Unified Modeling Language, Lenguaje de Modelado Unificado.

1. Introducción

1.1. Contexto

La seguridad, en la actualidad un sector muy importante en el que se están realizando diversos estudios para su mejora en todas las aplicaciones de la sociedad. En especial en automoción en los que en los últimos años se están introduciendo significativas mejoras a los vehículos para que tengan una máxima seguridad a la hora de la conducción incorporando sistemas obligatorios en cada uno de los vehículos independientemente de su forma, tamaño o aplicación.

Una de las medidas que ha cobrado relevancia en los últimos tiempos son sistemas de seguridad integrados en los vehículos autónomos, como son las setas de emergencia. Estos dispositivos tienen una funcionalidad clara, actuar en situaciones de peligro, pudiendo cortar la alimentación eléctrica, parando completamente el vehículo o parando el motor entre otras funciones.

En este proyecto se realizará un sistema de setas de emergencia integrado en un vehículo autónomo con el objetivo de actuar en situaciones de peligro, cortando el control autónomo del sistema o en cuyo caso haciendo una parada de emergencia del vehículo. Si bien, el objetivo está definido, siempre es posible su cambio, ya que la seta de emergencia envía un mensaje a la centralita del vehículo, pudiendo actuar en diferentes ramas del sistema como activar algún sistema de emergencia si fuese necesario.

Si bien no es común encontrar una seta de emergencia como tal en el habitáculo de un vehículo convencional, es interesante observar que se encuentra presente en vehículos agrarios, autobuses, trenes, vehículos aéreos, marítimos, vehículos destinados a pruebas y vehículos destinados a la competición. Sin embargo, esto no significa que no tenga sentido integrar uno de estos sistemas a un vehículo convencional ya que añadiría un extra de seguridad como si fuera un actuador más.

Con este proyecto se busca la mejora de la seguridad en la carretera y otros ambientes de conducción autónomos mediante la implementación de una seta de emergencia. Se espera que con este proyecto se aclare varios aspectos relacionados con la integración del sistema de emergencia en vehículos convencionales y destinados a pruebas. En primer lugar, se anticipa que la implementación de este dispositivo genere un avance significativo en términos de seguridad vial, al proporcionar a los conductores una herramienta adicional para hacer frentes a situaciones de emergencia de manera más efectiva.

Además, se espera identificar beneficios como la reducción del tiempo de respuesta ante incidentes viales y la minimización de los daños en el vehículo. Asimismo, se anticipa que este sistema pueda contribuir a la reducción de accidentes causados por fallos mecánicos o eléctricos pudiendo actuar en cuanto surja algún tipo de problema en cualquier situación, lo que tendría un impacto positivo en la seguridad de los ocupantes del vehículo y de otros usuarios de la vía.

Por tanto, se explorará en detalle la viabilidad técnica y los posibles beneficios de integrar este sistema, con el objetivo de contribuir al avance en los campos de los vehículos autónomos, la seguridad automotriz y proteger la vida de los conductores y pasajeros en los vehículos modernos.

1.2. Objetivos

El objetivo principal del sistema es la creación de un sistema de setas de emergencia conectadas a través de un bus CAN, en el que si se pulsa cualquiera de ellas se enviará el mensaje pertinente que convocará en una parada en el vehículo actuando en la mecánica correspondiente del vehículo.

Para llevar a cabo este proyecto se ha dividido el trabajo en diferentes objetivos parciales que se han realizado a lo largo del proyecto, si bien no todos han sido en un principio determinados, a lo largo del desarrollo se ha decidido integrar diferentes metas que finalmente se han podido llevar a cabo.

A continuación, se detallarán los objetivos que se han ido desarrollando para tener el sistema completo:

Diseño e impresión del sistema en PCB. Se ha implementado el sistema en una placa PCB con el objetivo de ser un producto robusto de calidad y ajustándose de forma que se adapte a todos las funcionalidades que incorporará, alejándose de los primeros prototipos en protoboard que se desarrolló al principio del proyecto.

Envío y recepción de mensajes a un ordenador de a bordo o centralita mediante el protocolo CAN. El sistema será capaz de enviar el estado actual cuando se realiza una determinada acción. Cuando es pulsada la seta de emergencia esta enviará un mensaje con la información de que se ha pulsado. De la misma forma, con los demás estados que se explicarán en la sección 4.1.

Simulación de varias setas de emergencia. Con el objetivo de poseer varias setas en el sistema, se ha desarrollado un microSwitch capaz de seleccionar una seta de entre 16. Por tanto, en el sistema desarrollado se seleccionará una de esas setas pudiendo intercambiar de manera fácil e intuitiva.

Integración de un buzzer creando un aviso sonoro al usuario. Para el correcto funcionamiento de esta sección, se ha desarrollado un divisor de frecuencia con una serie de resistencias, un potenciómetro y un transistor que es capaz de regular el sonido del buzzer, todo integrado en la propia placa.

Incorporación de un sistema de LEDs para la fácil identificación del estado de la seta. Esta sección se ha conseguido integrando un switch con un diodo LED integrado, el cual avisa de cuándo el sistema está armado o desarmado. Además, se disponen de diferentes LEDs para comprobar el correcto funcionamiento del sistema, como el LED de alimentación y el LED de señal para el buzzer.

Integración del sistema en una estructura sólida. Capaz de soportar altas temperaturas y fuertes golpes que pueden suceder en situaciones de peligro. Integrado en una estructura metálica modificada a medida para la integración de la placa y sus respectivos actuadores.

1.3. Motivación

Este proyecto surge de un gran interés sobre los sistemas Hardware y Software que llevan integrados los vehículos en la actualidad, además del que han proporcionado diferentes asignaturas cursadas a lo largo de la carrera, como sistemas empotrados y sistemas en tiempo real con el estudio de impresión de placas y tratamiento de la información con el uso de interrupciones.

Además, debido a un proyecto que surgió en el Instituto Universitario de Investigación del Automóvil (INSIA), el cual se quería desarrollar un sistema de seguridad extra en uno de sus prototipos autónomos, se decidió incorporar una seta de emergencia integrada en el vehículo.

A todo esto, se le suma el gran potencial que tienen los sistemas de seguridad en la actualidad, siendo muy importante tanto en la vida profesional como en el día a día en todo tipo de vehículos y la facilidad que tienen las grandes incorporaciones del desarrollo de este tipo de sistemas, teniendo la posibilidad de integrar este proyecto en un entorno profesional.

No hay que dejar de lado la motivación que genera realizar un proyecto a medida, utilizando programas Software para el diseño de la placa, y la utilización de maquinaria para las perforaciones que se han realizado para la correcta integración de la placa con la estructura utilizada.

2. Estado del arte

En esta sección se tratará la evolución de la seguridad en los vehículos, así como las diferentes tecnologías que han surgido a lo largo del tiempo tanto en el apartado industrial como en el cotidiano [1].

2.1. Historia de la seguridad en los vehículos

En este apartado se expondrán las actualizaciones que ha tenido la seguridad vehicular a lo largo de la historia

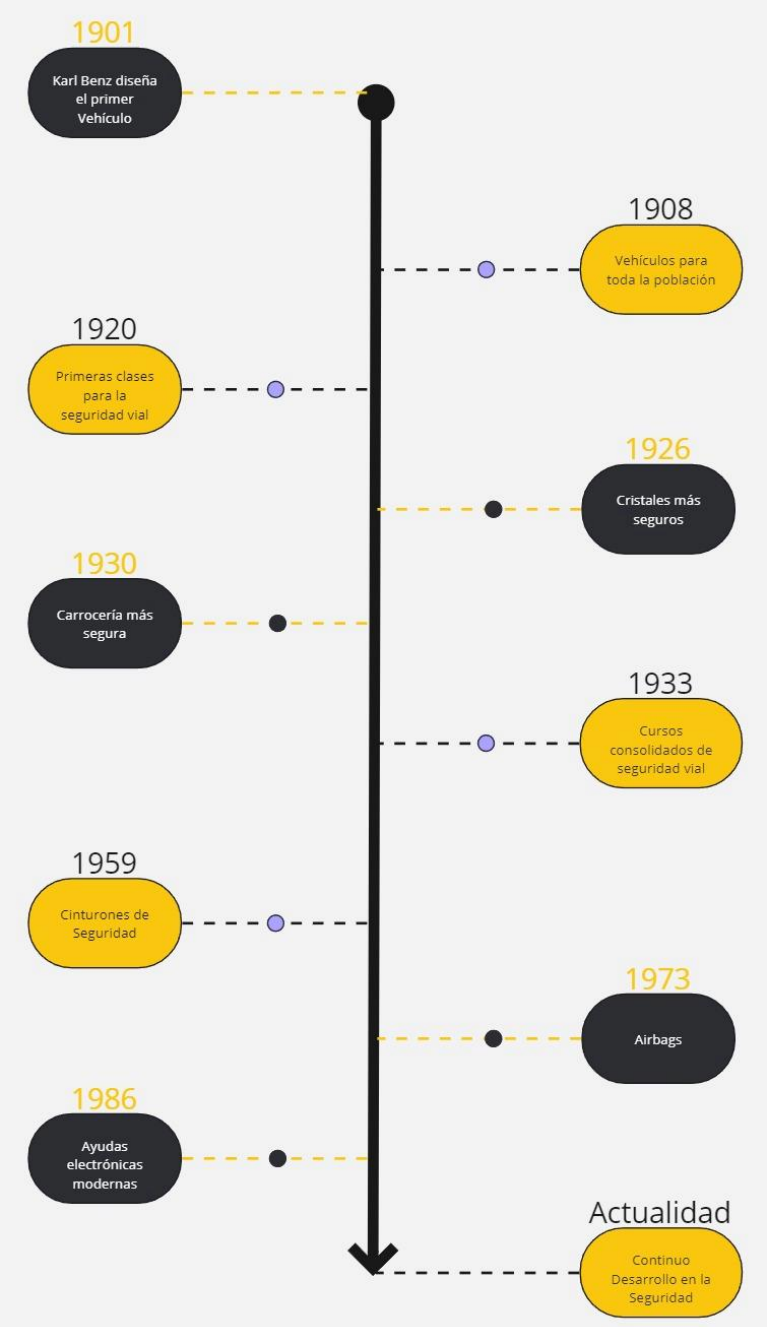


Figura 1. La línea de tiempo de la seguridad

2.2. Los principios

El automóvil moderno fue creado por el alemán Karl Benz (quien sería famoso por presentar el Mercedes en 1901, lo cual desembocaría en la conocida marca Mercedes-Benz) en Estados Unidos. Sin embargo, estos nuevos medios de transporte no estuvieron disponibles para la población hasta que Henry Ford introdujo su Modelo T en 1908. Las prácticas de producción eficientes de Ford, incluida su famosa línea de montaje, permitieron reducir el costo de los automóviles. Esto convocó que numerosas familias estadounidenses se permitiesen disfrutas de esta nueva tecnología. Sin embargo, pronto surgieron claros peligros. El exceso de velocidad, la conducción imprudente, la colisiones y numerosas muertes de peatones fueron un tema a solucionar a lo largo de 1910.

2.3. El Antiguo Conductor

En 1913 el Consejo Nacional de Seguridad estableció una misión para eliminar las muertes laborales y en hogares, entre los que se encuentran las prevenientes en el camino. Se enfocó en el factor humano en los accidentes en carretera, en la que se unieron otros grupos de defensa, difundiendo una campaña en iniciativa a la seguridad.

La industria automotriz estaba convencida de que los automóviles no contribuían en los accidentes si estos se conducían de una manera correcta, dato interesante ya que tiene similitud con el sector de los automóviles autónomos, ya que son muchos quienes afirman que los accidentes son causados por un factor humano.

Como consecuencia a esos nuevos sucesos se crearon diversos cursos que se ofrecían a la población para reducir la tasa de accidentes. Es en 1920 cuando los distritos escolares de todo el país comenzaron a ofrecer instrucciones sobre la seguridad vial, en concreto en 1927 Gilbert Minnesota, se convocó el primer curso de educación para conductores.
Seguidamente en 1933, Amos Neyhart quien más adelante será conocido como el padre de la educación para los conductores [2], comenzó a ofrecer cursos de seguridad vial. Tal fue el éxito de los primeros cursos, que para 1936 ya se ofrecían clases en la Universidad del estado de Pensilvania.

2.4. Soluciones Tecnológicas

Pronto, la industria automotriz se dio cuenta de que los vehículos necesitaban unas características de seguridad, estas podrían prevenir colisiones y reducir la gravedad de las lesiones sufridas en accidentes.

- **Cristal más seguro**
  Los primeros automóviles montaban parabrisas y ventanas de vidrio plano, en la que en una colisión se rompen en piezas afiladas que pueden herir de gravedad tanto a los conductores como a los transeúntes. Es el sedán Stutz el cual fue el primero en incorporar cristales más seguros en 1926.

Figura 2. Sedán Stutz

Este vehículo incorpora cables horizontales en el vidrio para ayudar a controlar la rotura, aunque esta solución no fue muy duradera. Pronto los fabricantes de vehículos tuvieron una solución más efectiva fabricando un cristal de vidrio y celuloide que mantenía los fragmentos unidos en el impacto. Aunque no esta no fue la única solución al problema. En el Ford Modelo A de 1928 incorporaba el vídeo triplex facilitando la comercialización en masa, Duplate era una técnica que consistía en dos láminas de vidrio con una capa intermedia de celuloide que incorporaban los Cadillac en 1930.
Hoy en día se utiliza un vidrio de tipo laminado, dando una gran resistencia gracias al tipo de material usado ya que es un termoplástico transparente [3].

- **Carrocería más segura**
  En 1930 el mundo del automóvil tendió a líneas más continuistas y modernas ya que esto es lo que les gustaban a los compradores. Los chasis se hicieron más aerodinámicos que hicieron que el mundo del automóvil fuese un reflejo de la vida moderna y la tecnología. Estos chasis además de ser más vistosos tenían una estructura más rígida que les ofrecía una mayor resistencia en caso de accidente.

  Los salpicaderos y los accesorios que presentaban podían causar lesiones muy graves en los tripulantes, como lesiones craneales y faciales. Desde 1950 se apostó por salpicaderos acolchados que ayudasen a reducir las lesiones causadas por estos adornos.

Figura 3. 1950 Delahaye 135M Interior

En la anterior Figura se puede ver el interior del Delahaye 135 M, uno de los vehículos de los años 50 que cuentan con un salpicadero no seguro con las regulaciones actuales, presentando diferentes ornamentos que pueden causar lesiones graves en caso de accidentes.

- **Cinturón de Seguridad**

  El cinturón moderno se seguridad fue inventado en 1959 por Nils Bohlin, un ingeniero sueco que trabajaba para Volvo. La mayoría de los automóviles tenían cinturón de seguridad de dos puntos que se ataban al abdomen como se muestra en la Figura 4, estos podían causar lesiones internas a los pasajeros.

Figura 4. Cinturón de Seguridad de 2 puntos

La Administración Nacional de Seguridad de Tráfico en las Carreteras estima que estas medidas de seguridad salvan más de 11 mil vidas cada año solo en los Estados Unidos. National Highway Traffic Safety Administration ***(NHTSA)*** afirma que los cinturones de seguridad ayudan en la reducción de daños personales en un 87%, contrastado por datos en el 2015

En la Figura 5 se puede observar los cinturones de seguridad de 3 puntos usados en la actualidad.

Figura 5. Cinturón de Seguridad de 3 puntos

- **Airbags**
  Aunque fueron inventados en 1952 por John W.Hetrick [4], quien presentó la patente en los Estados Unidos, el uso de esta medida de seguridad no fue hasta los principios de los 70. El primer vehículo que incorporaba estos dispositivos fue el Oldsmobile Toronado en 1973 mostrado en la Figura 6. El primer Airbag, aunque no fue bien acogido en el mercado, habiendo disputas negando su validez. Fue en los años 80 cuando esta medida se extendió y se convirtió en uno de los estándares de seguridad más importante en el mundo de la automoción.

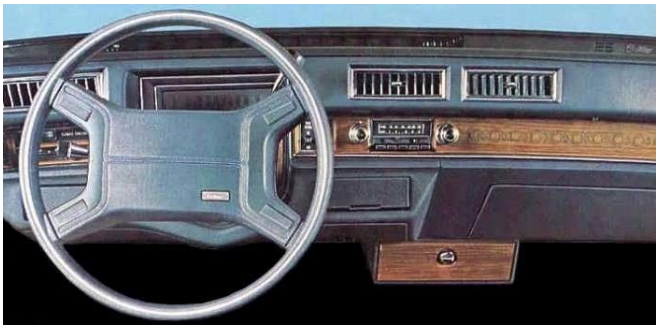


Figura 6. El primer Airbag

- **Asientos más seguros y cómodos**
  Los primeros asientos no eran muy diferentes a sillas de montar, sin embargo, con el aumento del tiempo de conducción y la creciente seguridad que se estaba implementando en los vehículos, la tendencia fue a asientos modulares, los cueles pueden ser abatibles pudiendo ganar espacio en el habitáculo, además de añadir más relleno en los costales para

que en el momento de tomar una curva genere más sujeción al ocupante.

Figura 7. Asientos abatibles

- **Ayudas electrónicas, Control de Tracción ESP**
  Diseñado por Bosh, Surgió en 1995, su función es la de ayudar al conductor en caso de que pierda el control del vehículo, para eso, este sistema controla varios parámetros electrónicos a través de varios sensores y actuando directamente en la rueda conveniente acelerándola o frenándola convenientemente para no perder la adherencia en curva en función del ángulo de giro del volante.

  El primer vehículo que incorporó el ESP fue el Mercedes-Benz A 140, y tras comprobar su buen funcionamiento, la Unión Europea decidió incluirlo de forma obligatoria en todos los nuevos vehículos a partir del 1 de noviembre de 2014.

Figura 8. Control de Tracción ESP

2.5. Comunicación entre sistemas en el vehículo

Debido al incremento de sensores y actuadores en los vehículos, se hizo inviable controlar cada uno de ellos de forma individual con un cable o varios, es por eso por lo que se creó un sistema de comunicación interna. Es así como se pudo centralizar el control de estos elementos permitiendo reducir la cantidad de cable y aumentar el control de lo que ocurre en el vehículo. Una posible distribución de un sistema de comunicación es el sistema CAN con sus centralitas correspondientes, tal como se puede ver en la Figura 9.

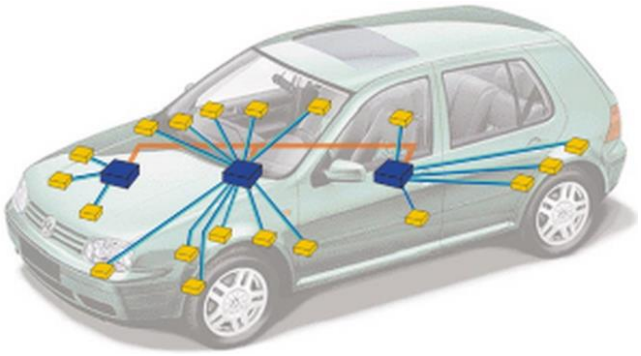


Figura 9. Sistema CAN

2.5.1. Bus CAN

Controller Area Network (CAN) es un protocolo de comunicación en serie desarrollado por Bosh en 1983, con el objetivo de intercambio de datos entre ECUs. Esto permite reducir significativamente el número de sensores y cables en el vehículo. Aunque no empezó a implementarse hasta 1992, con el vehículo que se muestra en la Figura 10, estandarizándose en 1993 bajo la ISO 11898, habiendo tenido las siguientes actualizaciones:

- *ISO 11898:1993*
- *ISO 11898-1:2003*
- *ISO 11898-2:2003*
- *ISO 11898-3:2006*
- *ISO 11898-4:2004*
- *ISO 11898-5:2007*
- *ISO 11898-6:2013*
- *ISO 11898-7:2015*

- *ISO 11898-8:2016*

Figura 10. Mercedes Clase E 1992, primer vehículo en incorporar el sistema CAN

Basado en la topología BUS en un medio multiplexado, explicado en este artículo de la universidad de Murcia para su asignatura de Sistemas Embebidos [5], el cual usa el protocolo serie asíncrono del tipo CSMA/CD. Es un protocolo Multicast, es decir, puede recibir y enviar datos simultáneamente. CSMA permite que cada nodo de la red debe monitorizar el bus, si detecta que no hay actividad es posible enviar el mensaje. CD permite detectar una colisión si dos nodos comienzan a transmitir al mismo tiempo.

Al seguir el sistema OSI (Open System Interconnection) le permite simplificar tareas de comunicación de actuadores y sensores de distintos fabricantes en una misma red. Por lo que facilita la conexión e interpretación de los datos.

Todo esto le permite obtener las siguientes propiedades [6]:

- Garantía de tiempos de Retardo
- Recepción múltiple con tiempos de sincronización
- Multimaestro
- Detección de error y señalización
- Retransmisión de mensajes erróneos en cuanto el bus esté disponible de nuevo
- Distinción de errores temporales y fallos permanentes
- Desconexión automática de nodos defectuosos
- Priorización del mensaje
- Seguro en sistemas con gran cantidad de datos

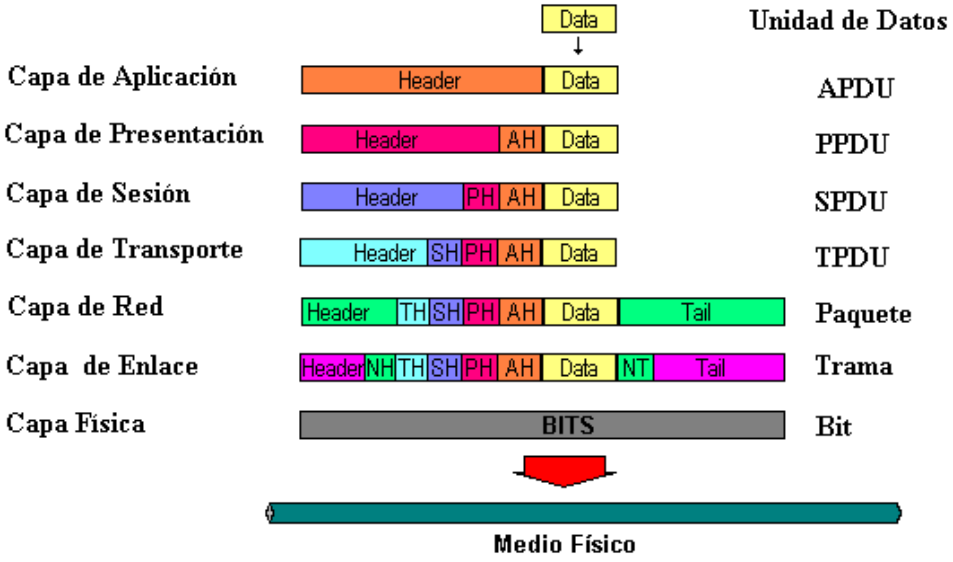


Figura 11. Modelo OSI

- **Capa de Enlace**

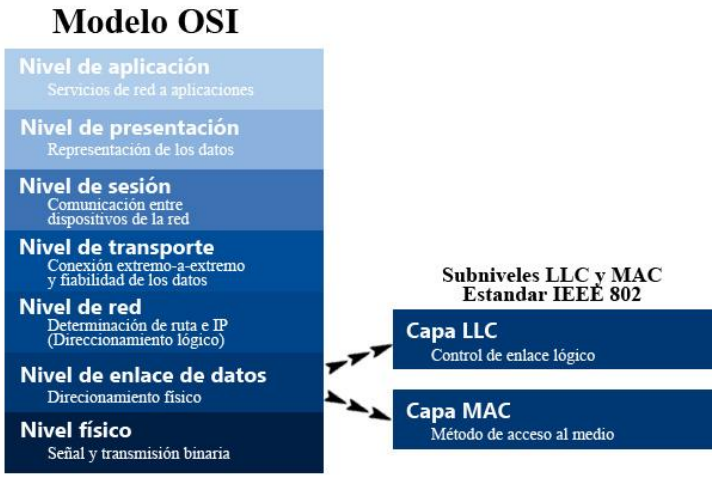


Figura 12. Modelo OSI Nivel de enlace de datos

- *Control de enlace lógico (LLC)*

  Esta capa es la encargada de la comunicación de los datos. Se encarga de filtrar los mensajes recibidos, asegurando que solo los mensajes relevantes sean procesados y transmitidos a capas superiores para su posterior tratamiento. Además de optimizar el uso del ancho de banda también reduce la carga de procesamiento entre dispositivos receptores

  Durante el proceso de transmisión y recepción de datos, la capa LLC proporciona una serie de servicios para una transmisión y recepción de datos eficiente y fiable.

La corrección de errores y la detección de estos, control de flujo para evitar la congestión de red y la segmentación y reensamblaje de largos mensajes ofrecen una transmisión óptima.

Decide qué mensajes recibidos de MAC se aceptan incluyendo la priorización de ciertos tipos de mensajes cortos sobre otros, en función de la importancia y la urgencia de la información transmitida

Proporciona Medios para el restablecimiento y notifica la sobrecarga del bus, lo que mantiene la integridad y eficiencia del sistema de comunicación

- *Control de Acceso al Medio (MAC)*

  Es la responsable de coordinar y gestionar el acceso al medio de comunicación compartida por los nodos de la red CAN. Se encarga de recibir los mensajes provenientes de otros nodos de la red y los envía a la subcapa de Control de Enlace Lógico (LLC). Además de aceptar los mensajes que le llegan desde LLC que ya estén listos para ser transmitidos.
  En esta subcapa, se llevan a cabo funciones para el correcto funcionamiento del protocolo CAN, incluyendo la estructuración de las tramas, el arbitraje para resolver conflictos de acceso al bus, el reconocimiento de mensajes recibidos correctamente, la detección y manejo de errores y la señalización de eventos importantes en la comunicación, es decir, decide si el bus está libre para comenzar una nueva transmisión o si la recepción acaba de empezar.

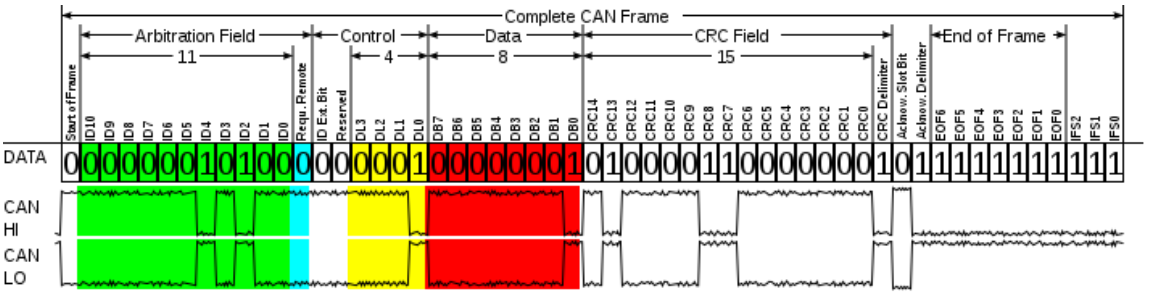


Figura 13. Trama CAN

- **Capa Física**

  Se encarga de las conexiones físicas de la red para la transmisión de datos entre nodos, especifica las características eléctricas de las señales, así como la codificación, decodificación, temporización y sincronización de los bits.

  - *Tipos*

    Existen dos tipos de redes CAN, la red CAN de alta velocidad (hasta 1Mbit/s) y la red CAN de baja velocidad tolerante a fallos (hasta
    125kbit/s)

CAN de alta velocidad

Bajo la ISO 11898-2, usa un único bus lineal en el que en los extremos presenta resistencias de 120Ω, que debe coincidir con la impedancia característica del bus, cuyo objetivo es evitar interferencias. Este CAN permite una velocidad máxima de hasta 1 Mbit/s

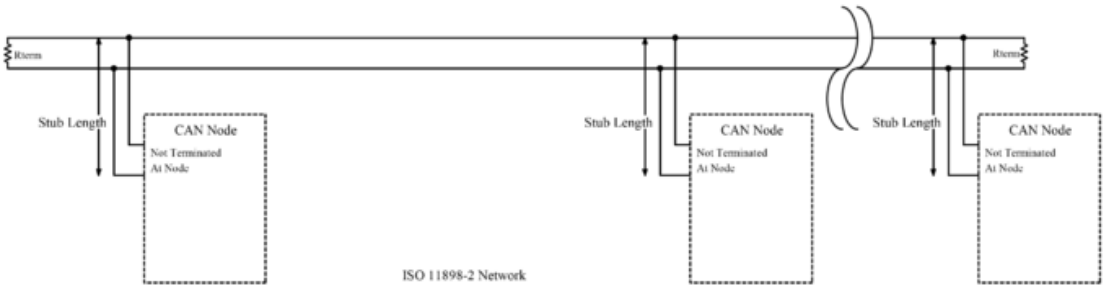


Figura 14. CAN de alta velocidad

CAN de baja velocidad tolerante a fallos

Bajo la ISO 11898-3, puede utilizar varios tipos de buses, como el bus lineal utilizado en el de alta velocidad, bus en estrella o múltiples buses en estrella conectados por un bus lineal.

Necesariamente tiene que estar terminado por cada nodo con una fracción de la resistencia de la terminación total, cuyo valor es próximo a 100Ω, pero no inferior. Este estándar permite una velocidad máxima de 125kbit/s.

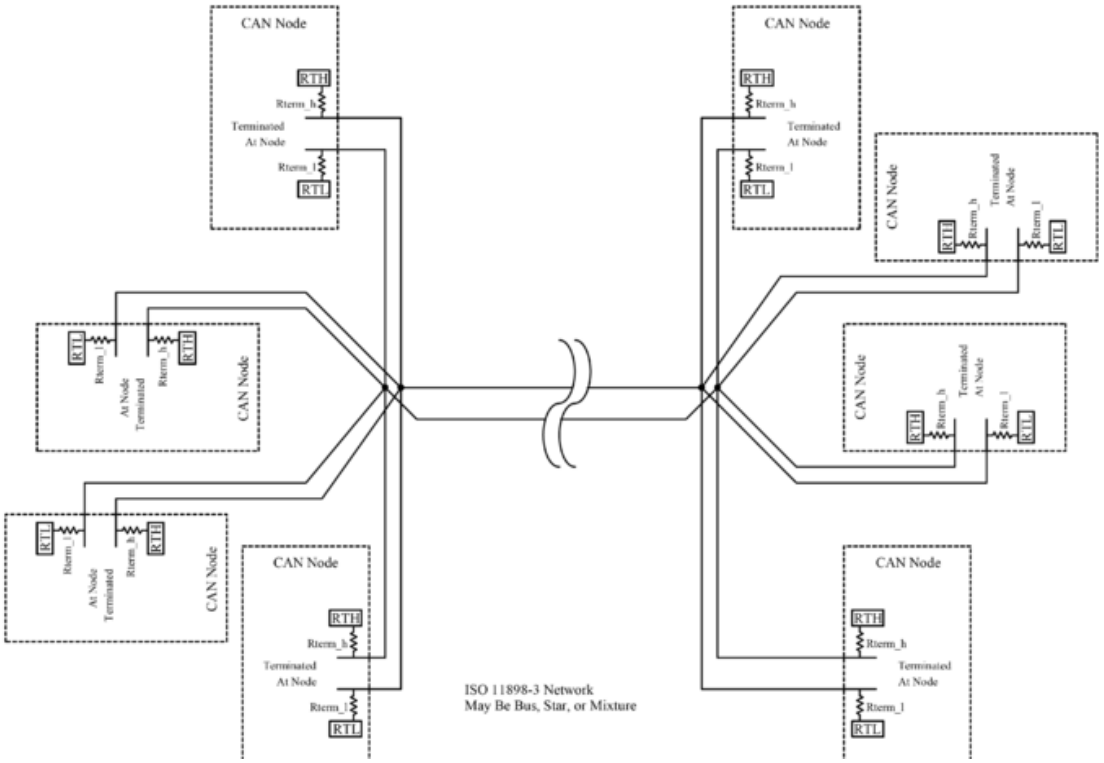


Figura 15. CAN de baja velocidad tolerante a fallos

- *Niveles de tensión del bus*

  Para llevar a cabo la transmisión de señales en el bus, presenta dos cables trenzados. Las señales se denominan CAN_H(CAN *high*) y CAN_L (CAN *low*). El bus tiene dos estados: estado dominante y estado recesivo.

  En el estado *recesivo*, los dos cables del bus se encuentran al mismo nivel de tensión, (lo que se denomina como *common-mode voltage*). En el estado *dominante* hay una diferencia de tensión entre CAN_H y CAN_L, de al menos 1.5V, lo que genera una transmisión de señal de tensión diferencial proporcionando protección frente a interferencias electromagnéticas.

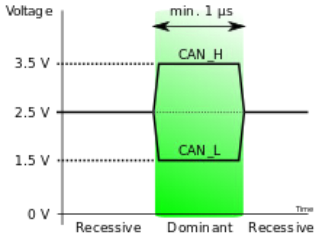


Figura 16. Niveles de tensión del bus CAN

Por tanto, la tensión en modo común puede estar en cualquier punto entre -2 y 7V. La diferencia entre CAN_H y CAN_L en modo *dominante* debe estar entre 1,5 y 3V. En cambio, no se especifica la tensión en modo común cuando el bus está en modo *recesivo*, deba estar comprendida entre la tensión de CAN_L y la tensión de CAN_H cuando el bus está en modo *dominante*. Esto permite la conexión directa entre nodos que operen a distintas tensiones, e incluso nodos que sufran diferencia de tensión entre sus respectivas tierras

- *Cable y conectores*

  Aunque en el estándar CAN no se especifica ningún tipo de conector para el bus, hay varios formatos que son utilizados y son comúnmente aceptados, como el conector D-sub de 9 pines (db-9), el cual presenta la señal CAN_L en el pin 2, tierra en el 3 y la señal CAN_H en el 7 y existiendo pines opcionales como el 9. El diagrama correspondiente se muestra a continuación.

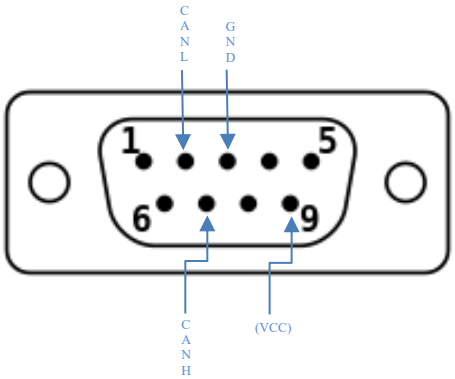


Figura 17. Conector db-9 CAN

- *Sincronización de bits*

  Los nodos de un bus CAN deben de trabajar con la misma tasa de transferencia nominal. El bus CAN no usa una señal de reloj separada lo que causa una diferencia entre la tasa de transferencia real de los distintos nodos. Es necesario un método de sincronización, importante en la fase de arbitraje ya que durante este cada nodo debe ser capaz de observar tanto los datos transmitidos por él como los transmitidos por los demás nodos.

  El requisito mínimo para un bus CAN es que dos nodos, estando cada uno en un extremo de la red con el máximo retarde de propagación entre ellos, y cuyos controladores CAN tienen unas frecuencias de reloj en los límites opuestos de la tolerancia de frecuencia especificada, sean capaces de recibir y leer correctamente todos los mensajes transmitidos.

  El controlador CAN espera que una transición del bus de recesivo a dominante ocurra en un determinado intervalo de tiempo. En el caso de que la transición no ocurra en ese intervalo, es el controlador quien reajusta la duración del siguiente bit. Ese ajuste se lleva a cabo dividiendo cada bit en intervalos o quantum de la forma:

  - *Segmento de sincronización (Sync)*
    Periodo de tiempo en el que ocurren las transiciones de recesivo a dominante, es decir, la transición en la que todos los nodos se encuentran al mismo nivel (*recesivo*) a donde hay una diferencia de tensión de al menos 1,5V (*dominante*).

  - Segmento de propagación (Prop)
    Es el tiempo que compensa los retardos de propagación a lo largo de la línea.

  - Segmentos de fase 1 y 2 (Phase 1, Phase 2)
    Son usados para la resincronización de los nodos. La fase 1 puede ser alargado o el 2 acortado para la resincronización

  - Punto de muestreo (Sample Point)
    Se encuentra inmediatamente después del segmento de fase 1 y se encuentra habitualmente cerca del 75% de la duración total del bit

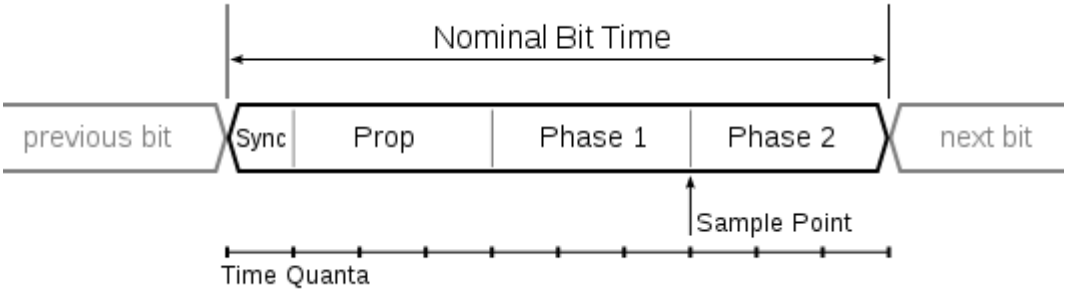


Figura 18. Ejemplo de cuantificación de un bit CAN con 10 cuantos de tiempo por bit

2.5.2. CANOpen

Es un protocolo de capa de aplicación CAN desarrollado por CiA (CAN in Automation) en 1995, en concreto trabaja en un protocolo de aplicación industrial de bajo nivel para aplicaciones de automatización. Tiene como objetivo conectar dispositivos de automatización entre sí mediante mensajes entre pares [7]. Aunque no es un protocolo como tal ya que sigue el estándar de comunicación física CAN ya explicado en el anterior apartado, es importante su explicación ya muchos dispositivos se basan en este protocolo.

- **Estructura de datos**

  CANopen utiliza el hardware de CAN para definir un protocolo de capa de aplicación que estructura la tarea de configuración, acceso y mensajería entre dispositivos. El diccionario de objetos es el punto de conexión entre la interfaz de comunicación y el programa, todos los objetos de comunicación se describen en el diccionario de objetos de forma estandarizada, pudiendo acceder a estos mediante un índice de 16 bits u 8 bits. Este diccionario proporciona objetos de comunicación estandarizados para datos en tiempo real (PDO), datos de configuración (SDO) y funciones especiales (como mensajes de sincronización y mensajes de emergencia)

  Existen mensajes que están incrustados en la capa de aplicación CAN (explícitos e implícitos) que permiten el acceso al diccionario de objetos. Los explícitos permiten a un dispositivo solicitar el valor de datos de un elemento en el diccionario de objetos o establecer un valor en el elemento de ese diccionario. Los mensajes implícitos permiten transferir datos predefinidos
  de un dispositivo a otro sin sobrecarga.

  Es importante aclarar que los dispositivos CANopen no son dispositivos estrictamente iguales ni dispositivos estrictamente maestros esclavos, es decir, es posible que un dispositivo CANopen sea un Maestro para otro dispositivo CANopen y le ordene que tome alguna acción, al mismo tiempo, puede ser un dispositivo esclavo para otro que desea ordenarlo que toma alguna acción, y al mismo tiempo, que pueda intercambiar datos de pares con otro dispositivo.

  Estas funciones son gracias al diccionario de objetos que organiza todas las comunicaciones y los datos expuestos a la red dentro de un mismo dispositivo. Si se permite el acceso, un dispositivo en una red CANopen puede configurar otro para realizar comunicaciones o aceptas mensajes, aunque los dispositivos CANopen Máster están predefinidos. En otras palabras, el diccionario de objetos CANopen es el núcleo de un dispositivo CANOpen y contiene los tipos de datos admitidos, los objetos de comunicación y los objetos específicos del proveedor.

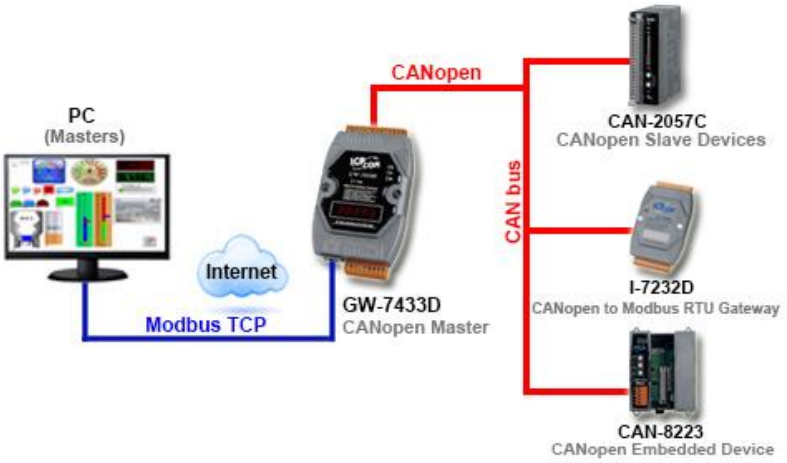


Figura 19. Protocolo CANopen

- **Mensajes y tipos de datos**

  Las transferencias de datos de PDO son de alta prioridad y no confirmadas que mueve la E/S y la información de estado de alta prioridad de un dispositivo a otro. Las PDO se inician internamente por el dispositivo o desde algún mensaje externo.

  La especificación CANopen define un conjunto de conexiones predefinidas, es decir, un conjunto de PDO bien entendidos y preconfigurados que, al utilizar el conjunto de conexión predefinido, no necesita pasar por el problema de configurar los PDO en ambos lados, asegurarse de que coincidan y probar las comunicaciones.

- **Objetos de datos de servicio**

  Admiten la transferencia de comandos y datos asíncronos entre los dispositivos CANopen. Los SDO existen para la configuración, aunque a algunos proveedores prefieren usarlos para la transferencia de datos. El destino del mensaje SDO, el Diccionario de objetos en el cual actúa, es el Servidor para el SDO y el iniciador el cliente del SDO.

  Los SDO presentan dos propósitos:

  - Leer una entrada del diccionario de objetos en otro dispositivo CANopen.
  - Escribir una entrada del Diccionario de objetos en otro dispositivo.

Cada SDO contiene un complemento completo de 8 bytes de datos, aunque a veces no se utilizan, a diferencia de los PDO, las transferencias siempre son confirmadas por el receptor y pueden transferir muchos más datos y tener varios tipos.

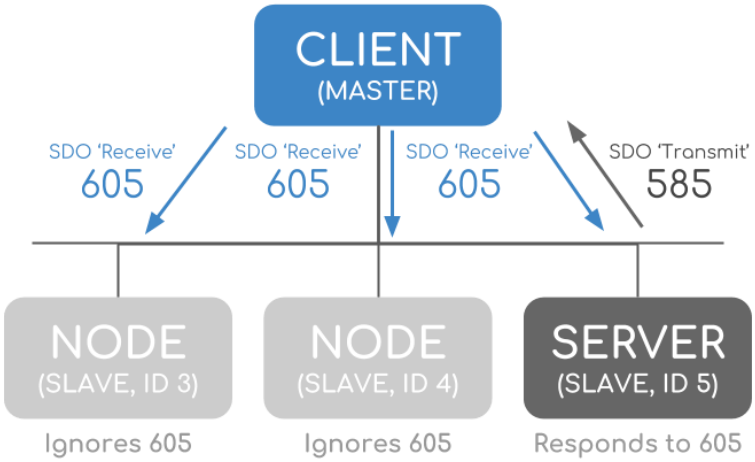


Figura 20. Objetos de Datos de Servicio (SDO)

- **Tipos de mensajes**

  - **Mensajes de Sincronización**
    Son un conjunto de mensajes emitidos por un dispositivo de tipo Maestro que sirve para sincronizar las acciones de un grupo de dispositivos CANopen. Se reserva una ID de conexión en el grupo de mayor prioridad (ID CAN más baja) para garantizar que el mensaje de sincronización se transmita de manera confiable a través de la red

  - **Mensajes NMT**
    Son un conjunto de mensajes que utilizan un maestro con un conjunto de dispositivos esclavos CANopen que implementan el conjunto de conexiones predefinidas. El protocolo NMT es implementado utilizando una solo trama CAN y se le asigna la ID del CAN de mayor prioridad. El comando NMT consiste en un byte de comando que indica la acción a realizar y una ID de nodo. Es común utilizar una ID de nodo de cero para indicar que todos los dispositivos en la red deben ejecutar el comando proporcionado.

  - **Mensajes de Emergencia**
    Es el conjunto de mensajes utilizados por un dispositivo CANopen para transmitir alguna condición de error. El protocolo de emergencia transmite el objeto de datos de emergencia una vez por cada error interno del dispositivo. No se transmitirán mensajes adicionales a no ser que se genere una condición de error interno de un dispositivo diferente.
    Normalmente, el dispositivo maestro CANopen es el único que consumirá el mensaje del protocolo de emergencia.

  - **Mensajes de HeartBeat**

Es una única trama que transmite periódicamente el estado actual de NMT del dispositivo. En la siguiente imagen se muestra el mensaje CAN que hace de HeartBeat en este mismo proyecto.

Data	Cycle Time	Count
00 00 00 00 00 00 00 02	405,2	1861

Figura 21. Mensaje HeartBeat en la Seta de Emergencia

2.5.3. BUS MOST

El bus MOST es un estándar de bus de datos que se destina a la interconexión de componentes multimedia como audio, video y datos de navegación, en automóviles y otros vehículos [8]. Fue desarrollado por una asociación de empresas automotrices y de tecnología en 1997. La diferencia con respecto a otros estándares de buses es que se basa en uno de fibra óptica. Esto permite al MOST un tráfico de datos superior que el del resto de buses. El primer vehículo en incorporar este estándar fue el BMW Serie 7, junto al lanzamiento del sistema iDrive en 2001.

La red MOST suele utilizar una topología en anillo, aunque existen configuraciones en forma de estrella y doble anillo para aplicaciones críticas que pueden llegar a incluir hasta 64 nodos o centralitas en el sector de la automoción.
Una característica importante es que es un sistema que facilita la adición o extracción de dispositivos en la red. Aun así, debe de existir un nodo que tenga el papel de *timing máster*, dedicado a alimentar el bus con estructuras de datos (*frames*) a la red, o que sirva como puerta de entrada (*gateway*) de datos. La cabecera del paquete sincroniza de forma repetitiva el resto de los nodos llamados *timing slaves*.

- **Velocidad de transmisión**

  El sistema MOST es capaz de transmitir datos multimedia en alta calidad, como audio, video y datos de anvegación a una velocidad de 150Mbit/s. Es una alta velocidad de transmisión que permita una producción sin interrupciones de contenido en los sistemas de entretenimiento, además el uso de fibra óptica en el sistema permite una transmisión de datos muy elevada sin interferencias electromagnéticas, lo que mejora la calidad de la señal y la fiabilidad de la comunicación

- **Componentes**

  - *Fuente de datos*
    Es el dispositivo que genera los datos multimedia, como un reproductor de DVD, un sistema de navegación o una radio.

  - *Unidad Óptica*
    Lee los datos de un disco óptico, como un CD o DVD, y los convierte en datos digitales para su transmisión
  - *Controlador de red*
    Gestiona el tráfico de datos en el bus MOST, y se encarga de la sincronización de dispositivos.

- *Procesador de señal digital* (DSP)
  Procesa los datos multimedia para su transmisión.

- *Transceptor*
  Se encarga de la conversión de la señal digital en una señal óptica, y viceversa.

- *Fibra óptica*
  Es el medio de transmisión utilizado por el bus MOST para la comunicación. Esta fibra es capaz de transmitir grandes cantidades de datos a una velocidad elevada, además de resistir a interferencias electromagnéticas.

- *Dispositivos de usuario*
  Son los que reciben los datos multimedia transmitido a través del bus MOST. Son las pantallas, altavoces, etc.

- **Versiones**
  Debido a las mayores exigencias en cuanto a ancho de banda multimedia, el estándar MOST ha ido evolucionando a lo largo del tiempo, aunque las versiones son muy parecidas, no son compatible entre sí, siendo la última versión la MOST150 presentada en 2007, en la que el paquete de datos se incrementa hasta 3072 bits, unas 6 veces mayor que la primera versión, además de incluir un canal Ethernet con un ancho de banda variable.

2.5.4. BUS FlexRay

Es un protocolo de comunicaciones para buses de datos en el automóvil desarrollado por el consorcio FlexRay entre 2000 y 2009 que ofrece requisitos de rendimiento de tolerancia a errores y determinismo de tiempo para aplicaciones *x-by-wire*, es decir, conducción por cable, dirección por cable, freno por cable, etc [9].

Este protocolo ofrece una serie de características por las que se considera más avanzado que otros como el CAN y el MOST.
FlexRay ofrece una alta transmisión de datos (10 Mbit/s), un comportamiento estimulado por factores temporales y una redundancia, seguridad y tolerancia de errores [10].

Aunque es un protocolo relativamente nuevo, aún está siendo revisado con objetivos de ofrecer una respuesta rápida a las discontinuidades de la carretera para lograr una conducción lo más suave posible. El primer vehículo en montar este protocolo es el BMW X5, aunque se espera el uso de esta tecnología a gran escala.

Una de las características que distinguen a FlexRay de redes tradicionales como Ethernet es su diseño de red o topología. FlexRay admite conexiones pasivas multipunto simples, así como redes en estrella para redes más complejas.

- **Detalles técnicos**

  - *Reloj*
    FlexRay consiste en un bus y procesadores, los cuales son los que dirigen las centralitas electrónicas. Cada centralita tiene un reloj independiente, cuya desviación no puede ser más de 0.15% del reloj de referencia, es decir, 300 ciclos de una centralita emisora pueden ser 299 o301 de una centralita receptora.

  - *Bits en el bus*
    Solo una centralita emite en el bus en cada momento, en el que cada bit se mantiene en el bus durante 8 ciclos de reloj. Cada receptor tiene un búfer de los 5 últimos ciclos.
    Los errores de transmisión en un único ciclo pueden afectar los resultados de los bits presentes en la frontera de la trama, pero no en el centro de un ciclo de 8 bits.

  - *Muestro de bits*
    El valor del bit se muestrea en el centro de una región de 8 bits. Los errores se desplazan a los extremos, y el reloj se sincroniza frecuentemente para evitar desfases.

  - *Estructura*
    La comunicación se envía en estructuras soporte o *frames*. Si no hay ningún mensaje en el bus, el valor es igual a 1 (voltaje ato), y cada receptor sabe que la comunicación comienza cuando el voltaje cambie a 0.

- **El ciclo de la comunicación**
  Es el elemento fundamental del esquema de acceso a los medios. La duración de un ciclo se fija cuando se diseña la red, pero normalmente oscila entre 1 y 5 ms. Hay cuatro partes principales en un ciclo de comunicación, como se muestran en la Figura 22.

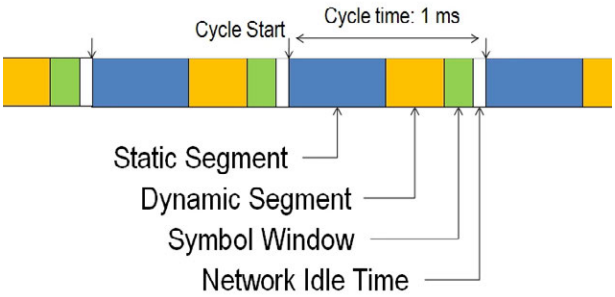


Figura 22. Bus FlexRay - Ciclo de comunicación

  - Segmento estático *(Static Segment)*
    Son espacios reservados para datos deterministas que llegan en un período fijo.

  - Segmento dinámico *(Dynamic Segment)*
    Se comporta de manera similar a CAN y se utiliza para una variedad más amplia de datos que están basados en eventos y no requieren determinismo.

- Ventana de símbolos *(Symbol Window)*
  Normalmente se utiliza para el mantenimiento de la red y la señalización para iniciar la red.

- Tiempo de inactividad de la red *(Network Idle Time)*
  Es un tiempo en el que se está en reposo, y es utilizado para mantener la sincronización entre los relojes de los nodos.

2.5.5. On Board Diagnostics (OBD)

El Sistema OBD facilita el diagnóstico de averías en un vehículo, este protocolo de control está regulado por la ley y es obligatorio en todos los vehículos.

Fue desarrollado en 1988 en California, Estados Unidos [11]. Para todos los coches de gasolina con el fin de controlar y reducir las emisiones contaminantes. En estos momentos, le método de diagnosis se centraba en el autocontrol, de manera que se marcaban unos límites máximos de emisiones para los vehículos. El sistema OBD controla este valor y cuándo se elevaba por encima de lo estipulado, avisa. En cualquier momento, se puede revisar y conocer los niveles de emisiones conectándose mediante dispositivos de mando a la central de diagnóstico.

En 1996 las medidas de control de emisiones de CO_2 se vuelven más estrictas y se actualiza a OBD 2 como obligatorio en todos los automóviles de Estados Unidos. En Europa este estándar es el EOBD [12] y en Japón el JOBD, aunque los vehículos pesados poseen una norma diferente, regulada por la SAE, conocida como J1939.

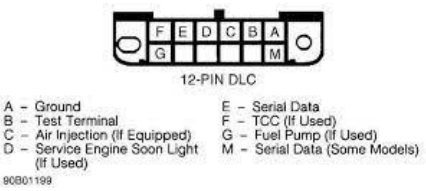
- **Funcionamiento**

  Los vehículos que incorporan un sistema OBD tienen una toma exterior en algún lugar, aunque suele ser dentro del habitáculo y cerca de la zona de los fusibles, bajo el volante o en la puerta del copiloto. El conector OBD del coche se debe conectar a un cable que a su vez se comunica con una central de diagnóstico.

  Tradicionalmente, la conexión del OBD ha sido mediante cableado, pero las conexiones bluetooth y wifi se han sido integradas y ya hay vehículos que mediante estos protocolos se conectan a diferentes dispositivos externos como puede ser un ordenador, tableta o smartphone para ejecutar la diagnosis.

- **Evolución**

  - *OBD-I*
    Fue la primera generación de regulación de OBD, obligaba a los productores a



instalar un sistema de monitorización de diferentes componentes cuya función era la emisión en automóviles. Fue obligatoria en todos los vehículos a partir de 1991. Estos sistemas no eran tan efectivos porque solamente monitorizaban algunos de los componentes relacionados con las emisiones y no eran calibrados para un nivel específico. El diagrama es el siguiente:

- *OBD-II*

  Es la segunda generación del sistema de diagnóstico a bordo y sucesor de OBD 1. Tiene como función alertar al conductor cuando el nivel de emisiones es 1.5 mayor a las definidas [13]. A diferencia del OBD1, OBD2 detecta fallos eléctricos, químicos y mecánicos que pueden afectar de mayor o menor medida al funcionamiento del vehículo.

Figura 23. OBD1

Para ofrecer la mayor información posible, guarda un registro de fallos y condiciones en la que ocurrió el error en el que cada uno tiene un código asignado.Al igual que OBD1, suele encontrarse cerca de los fusibles del vehículo, consola central o cerca de la puerta del copiloto y la funcionalidad de sus pines se muestra en la Figura 24

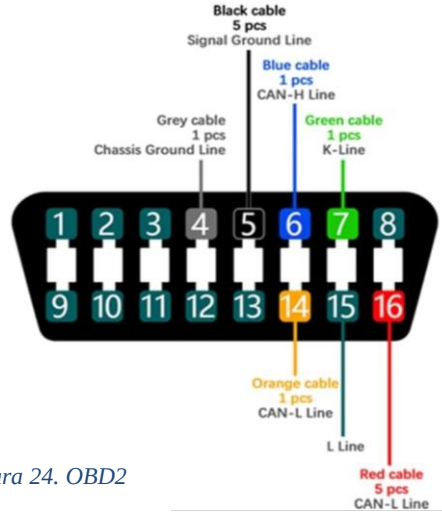


Figura 24. OBD2

- *EOBD* (European on Board Diagnostics)

  EOBD es un sistema mucho más sofisticado que OBD2 ya que usa mapas de las entradas a los sensores basados en las condiciones de operación del motor, y los componentes se adaptan al sistema calibrándose según datos anteriores.
  Una de las diferencias al OBD2 es que no se monitorizan las evaporaciones del depósito de combustible.
  Al igual que el OBD2, el conector tiene 16 pines y cada fabricante tiene una combinación específica. Dentro del estándar se pueden encontrar 5 comunicaciones de pinout y cada una tiene un protocolo específico de comunicaciones:

  - *J1850 VPW*
  - *ISO 9141-2*

- *J1850 PWM*
- *ISO 14230*
- *CANBUS*

2.6. Comunicación V2X

En esta sección se tratarán la nueva tecnología que está desarrollando y que ayuda en la seguridad vial, en concreto se centrará en las comunicaciones de vehículo-infraestructura (V2I) y vehículo-vehículo (V2V), es decir, los datos que envía el vehículo a diferentes dispositivos o infraestructuras situadas en las calzadas y carreteras para prevenir accidentes. Si bien esta tecnología no se ve en la vida cotidiana, tiene ya aplicaciones prácticas y será muy importante en la movilidad del futuro y la evolución del coche autónomo tal como se explica en el artículo de *tomorrow.bio* [14] y *race.es* [15].

Gracias a estas tecnologías y la transmisión de forma inalámbrica, se puede conocer el estado del tráfico, así como si ha sucedido algún accidente, facilitando a los conductores la toma de decisiones y reducir riesgos aumentando la seguridad de la conducción. La introducción del 5G hace posible que este intercambio de datos sea casi en tiempo real y pueda actualizarse al instante.

Para que la comunicación V2X (de ahora en adelante significando) tenga lugar, es necesario una sólida base tecnológica. Dos componentes son clave para esta comunicación: *Internet de las cosas (IoT)* y la velocidad de transmisión de datos.

- **Función del IoT**

  El IoT se refiere a la interconexión de dispositivos a través de internet y desempeza una función muy importante a la hora de la comunicación V2V y V2I. Al conectar dispositivos, el IoT crea la posibilidad de intercambiar datos sin fisuras, permitiendo que se mejore la toma de decisiones y la coordinación en la carretera. Esta conectividad no solo reúne semáforos, señales de tráfico, coches, etc. Si no que incorpora sistemas de respuesta a emergencias y medidas de seguridad para peatones.

  Gracias al IoT y esta tecnología un vehículo es capaz de comunicarse con otros vehículos próximos compartir información de su velocidad actual, su ubicación y hacia dónde se dirige y cualquier peligro que pueda haber encontrado, al mismo tiempo que puede estar recibiendo

  información de otros vehículos. Esto permite anticiparse y responder a peligros en tiempo real, como reducir la velocidad automáticamente si el ordenador de a bordo sabe que el siguiente semáforo se va a poner en rojo, indicarte una ruta mejor hacia tu destino debido a que alguna carretera está cortada, avisar a otros vehículos que ha habido un accidente en cierta ubicación, etc.

  Como se ha dicho antes, no solo abarca en las proximidades del vehículo, sino que también extiende su alcance proporcionando importantes datos como avisar a los servicios de emergencia si ha ocurrido algún accidente para que se actúe lo más rápido posible.

- **La importancia de la velocidad de transmisión de los datos**

  La velocidad de transmisión es crucial, ya que un retraso de incluso una fracción de segundo puede ser la diferencia entre evitar una colisión y un desastre potencial. La capacidad de transmitir datos de forma segura y rápida sin ninguna pérdida de información hace que haya una comunicación casi en tiempo real, minimizando la latencia y garantizando una rápida respuesta ante posibles peligros

  Uno de los factores clave para que estas tecnologías funcionen correctamente es el uso de la comunicación dedicada de corto alcance (DSRC) simulada en la Figura 25. Esta opera en la banda de frecuencia de 5,9GHz y permite el intercambio de información entre vehículos e infraestructuras en un rango de hasta 300 metros. Esto permite a los vehículos enviar y recibir datos a gran velocidad, garantizando que la información llega a su destino sin retrasos significativos.

Figura 25. V2V, V2I

Los grandes avances en la comunicación inalámbrica como el 5G, mejoran más la velocidad de transmisión de datos y gracias a su baja latencia y mayor ancho de banda, se permite intercambiar grandes cantidades de datos en tiempo real. Esto es muy importante a la hora de tomar decisiones en un muy corto período de tiempo, es decir, cuando el vehículo se acerca a una intersección o se encuentra con un obstáculo repentino en la carretera.

- **Implicaciones para la seguridad**

  Una de las aplicaciones más importante del V2V y V2I es en la seguridad en la carretera. Al mejorar la comunicación entre los vehículos e infraestructuras, es posible aplicar una será de medidas de seguridad, reduciendo significativamente accidentes en las carreteras.

  - *Gestión del tráfico y prevención de colisiones*

    Gracias a la capacidad de intercambiar información sobre la velocidad, dirección y distancia, los vehículos próximos pueden anticipar colisiones y tomar medidas preventivas. Esta interacción en tiempo real permite a los vehículos responder con rapidez, accionando

los frenos o ajustando su dirección para evitar colisiones.

La comunicación V2I contribuye a optimizar el flujo de tráfico al suministrar datos en tiempo real sobre la congestión. Al recopilar la información sobre patrones de tráfico, el estado de las vías y el volumen de los vehículos, las autoridades pueden tomar decisiones de manera más efectiva. Este enfoca reduce la probabilidad de accidentes ocasionados por la congestión y fomenta desplazamientos más fluidos y seguros para todos los usuarios.

- *Mejora de respuestas de emergencias*

  La comunicación V2V y V2I puede mejorar significativamente los esfuerzos de respuesta ante emergencias. La transmisión de datos entre vehículos y servicios de emergencia permite reducir los tiempos de respuesta y enviar rápidamente la ayuda necesaria.

  En el caso de que ocurra un accidente, los vehículos cercanos pueden alertar de inmediate a los servicios de emergencia, proporcionando la ubicación y gravedad del accidente. Estos datos que se transmiten en tiempo real permiten que el personal de emergencia actúa con rapidez y despliegue los recursos necesarios, lo que puede salvar vidas y minimizar los daños.

  Esta comunicación está diseñada con sólidas medidas de seguridad para proteger la privacidad e integridad de los datos transmitidos. Técnicas avanzadas de cifrado y protocolos de autenticación aseguran que solo entidades autorizadas puedan acceder a la información y utilizarla, evitando cualquier uso indebido o no autorizado.

- *Algunos problemas*

  La implantación del V2V y V2I requiere una inversión considerable en la infraestructura. Las redes viales existentes necesitan ser equipadas con dispositivos de comunicación adecuades para asegurar una conectividad constante y segura, lo implicaría un desembolso sustancial en este proyecto.

  Además, asegurar la compatibilidad entre vehículos y dispositivos es algo que puede generar diferentes problemas. La normalización y colaboración de fabricantes, proveedores de tecnología y organismos gubernamentales es un tema por tratar, por no mencionar las brechas de seguridad de información que esta tecnología puede presentar.

2.7. Elementos actuales en la seguridad vial

En este apartado se tratarán los sistemas actuales avanzados de asistencia a la conducción (ADAS). Una de las principales características de estos sistemas es la intervención automática sobre el freno o acelerador, dirección o señalización, entre otros, con el objetivo de evitar un accidente de tráfico y disminuir las lesiones sufridas [16].

La última actualización de ADAS obligatorias en España y dirigido por la DGT fue a día de hacer este documento fue el 6 de Julio de 2022 [17]. Estos nuevos sistemas son los que tienen que incluir los vehículos a partir de ese momento para que sean legales en España. Así, todos los vehículos de nueva homologación deben incorporar de serie al menos estos ocho sistemas: *Detector de somnolencia*

(DDR), Asistente de velocidad inteligente (ISA), Alerta de tráfico cruzado (RCTA), Caja negra (EDR), Alerta de cambio involuntario de carril (LDW), Sistema de frenado de emergencia (ESS), Inhibidor de arranque con alcoholímetro y Alerta de uso del cinturón en todas las plazas.

Para su correcto funcionamiento, estos sistemas necesitan un hardware específico. Por ejemplo:

- **Cámaras**
  Son utilizadas para detectar señales de tráfico, peatones, obstáculos y proporcionar funciones de asistencia a la conducción como el mantenimiento de carril y el frenado de emergencia.

- **Radares**
  Se utilizan para medir la distancia y la velocidad de objetos cercanos al vehículo, permitiendo funciones como el control de crucero adaptativo.

- **LIDAR**
  Utilizado para crear mapas detallados del entorno mediante el uso de pulsos láser, lo que permite una detección precisa de objetos y percepción del entorno 3D.

- **Sensores ultrasónicos**
  Proporcionan información sobre proximidad de objetos cercanos al vehículo, como otros vehículos, paredes y obstáculos. Mayoritariamente se utilizan para sensores de aparcamiento.

- **Actuadores**
  A parte de los sensores antes descritos, los actuadores son una parte fundamental de los sistemas ya que son los encargados de accionar diferentes partes del vehículo cuando sea necesario. Por ejemplo, los motores situados en los pedales de acelerador, freno y giro del volante.

A continuación, se explicarán los sistemas ADAS que ayudan más en la conducción de entre todos los que existen hasta el momento incluyendo en cada uno de ellos una figura representativa, diferenciando entre dos tipos: activos y pasivos.

Los sistemas activos son aquellos que intervienen directamente en el control del vehículo para prevenir accidentes o reducir su gravedad. Suelen utilizar sensores y actuadores para monitorear el entorno del vehículo y tomar decisiones en tiempo real.

Los sistemas pasivos actúan antes o después de que ocurra un accidente para reducir sus consecuencias y no intervienen en el control del vehículo. Están diseñados para proteger a los ocupantes en caso de colisión.

Sistemas Activos

- **Asistente de Velocidad (ISA)**
  Es un sistema de asistencia al conductor diseñado para ayudar a controlar y si es el caso limitar la velocidad del vehículo. Utiliza tecnología GPS y reconocimiento de señales de tráfico para

identificar los límites de velocidad en la carretera. El ISA alerta al conductor cuando se excede el límite de velocidad establecido, además de aplicar automáticamente los frenos o limitar la aceleración para mantenerse dentro de los límites.
El objetivo de este sistema es reducir el riesgo de accidentes relacionados con la velocidad excesiva. Su implementación puede variar dependiendo de las regulaciones locales y las preferencias del fabricante y del conductor. Este ADA tiene relación con el artículo publicado en el INSIA y escrito por Alberto Cruz entre otros [18].

Figura 26. Asistente de Velocidad

- **Cámara trasera con Detección de Tráfico Cruzado**

  Este sistema utiliza una cámara montada en la parte trasera del vehículo para detectar el tráfico que se aproxima desde los lados cuando se está realizando una maniobra de retroceso, como salir de un estacionamiento o ciertas maniobras.

  El sistema alerta al conductor de forma visual o auditiva sobre la presencia de vehículos o personas cercanos que podrían representar un riesgo de colisión. Esta tecnología mejora la seguridad al proporcionar al conductor información adicional sobre su entorno al conducir en marcha atrás, lo que ayuda a prevenir accidentes y minimizar daños.

Figura 27. Detección de Tráfico Cruzado

- **Alerta de Cambio Involuntario de Carril (LDW)**

LDW por sus siglas en Inglés Lane Departure Warning, diseñado para ayudar a prevenir salidas de carril no intencionadas. Utiliza sensores, cámaras o radares para monitorear la posición del vehículo dentro del carril en el que circula. Si detecta que el vehículo se está desviando involuntariamente de su carril sin que el conductor haya activado el intermitente, el sistema emite una advertencia visual, auditiva o táctil para alertar al conductor y ayudarlo a corregir su trayectoria. Esta tecnología resulta especialmente útil en situaciones de fatiga o distracción, ayudando a evitar accidentes por salidas de carril no intencionadas y mejorar la seguridad en carretera.

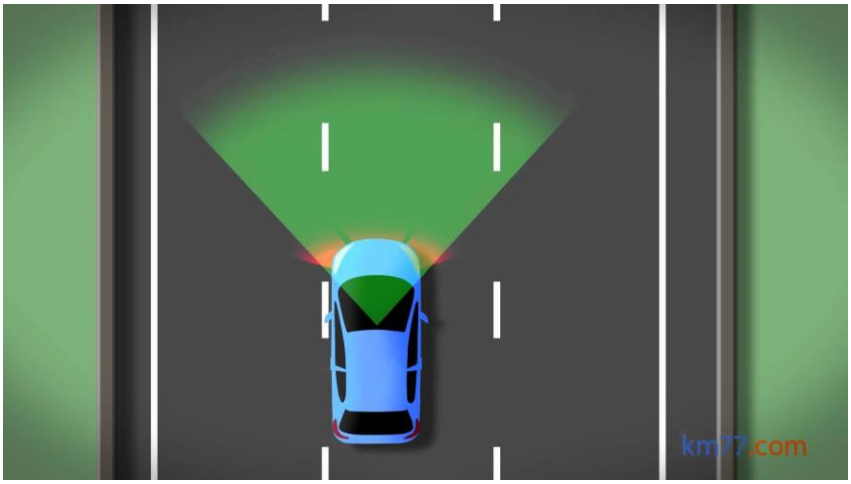


Figura 28. Alerta de Cambio Involuntario de Carril

- **Detector de Fatiga y Somnolencia**

  Como su nombre indica, este sistema está diseñado para monitorear el comportamiento del conductor y detectar signos de fatiga o somnolencia, utilizando sensores como cámaras integradas o sensores de movimiento, para analizar el comportamiento del conductor, incluyendo patrones de conducción, movimientos oculares y frecuencia del parpadeo.

Figura 29. Pantalla cona viso de detector de Fatiga

- **Sistema de Frenado de Emergencia BAS y EBA**

  Es un asistente desarrollado con el fin de acortar la distancia de frenado, está asociado al ABS y al ESP (Control de Tracción) [19]

- *BAS*: detecta situaciones de frenado rápido proporcionando una asistencia adicional para incrementar la fuerza de frenado aplicada por el conductor. Esto ayuda a reducir la distancia de frenado y detener el vehículo de manera más rápida.
- *EBA*: detecta cuando el conductor realiza una frenada de emergencia y automáticamente incrementa la presión de frenado, incluso si el conductor no aplica suficiente fuerza.

Ambos sistemas trabajan en conjunto para mejorar la capacidad de frenado como se pueden observar en la Figura 30.

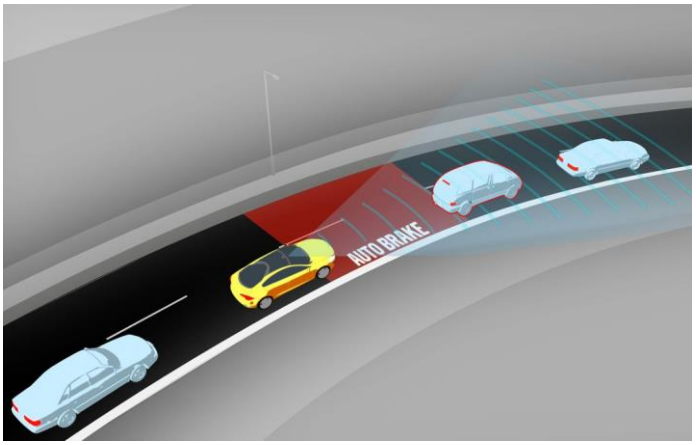


Figura 30. Los sistemas BAS y EBA

- **Detección de señales**

  Este sistema reconoce señales de todo tipo, incluidas las digitales fijas o variables y las restricciones temporales. No confundir con el asistente de Velocidad, ya que este ADA no solo muestra las señales de velocidad y no es capaz de cambiar la velocidad del vehículo, aunque puede funcionar en sintonía con el Control de Crucero Adaptativo (ACC).

- **Control de Crucero Adaptativo**

  El ACC (Control de Crucero Adaptativo) mantiene la velocidad programada de manera continua cuando está activado. Además, frena y acelera para adaptarse al tráfico y mantener una distancia preestablecida con el vehículo que le precede, incluso si éste se detiene por completo, además de mantener una distancia segura en situaciones de tráfico intenso. Ofrece confort y seguridad en viajes largos y puede funcionar en combinación con el Sistema de Detección de Señales, controlando la velocidad máxima de forma autónoma.

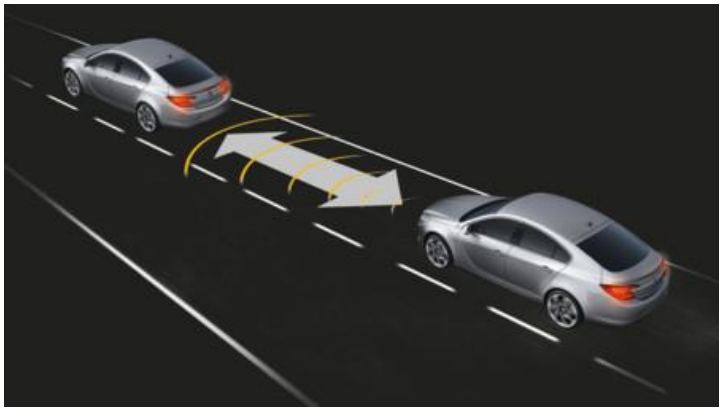


Figura 31. Control de Crucero Adaptativo Respetando las señales de tráfico

Sistemas Pasivos

- **Bloqueo del vehículo con alcoholímetro**

  Permite conocer al conductor su grado de alcoholemia y si está en condiciones plenas para

Figura 32. Alcoholímetro integrado en el vehículo

  conducir, además de impedir la conducción no permitiendo el arranque del vehículo si supera la tasa máxima de alcohol establecida. Está conectado al encendido del vehículo, de forma que, si el conductor no realiza el control no será posible arrancar, además de contar con un módulo de control que recoge y archiva los resultados para crear un historial.

- **Alerta de cinturón**

  Este aviso es uno de los asistentes electrónicos que la Unión Europea ha establecido que, a partir de 2022, todos los vehículos adopten obligatoriamente. Este sistema determina si una plaza está ocupada, mediante un sensor de peso, y comprueba mediante otro sensor situado en la hebilla del cinturón, que está bien cerrado. De esta forma, el conductor controlará que los pasajeros, se hayan puesto el cinturón correctamente.

  Si el vehículo recorre una distancia predeterminada, supero los 25km/h o transcurre más tiempo

del establecido sin que se haya colocado el cinturón, comenzará a emitir una señala acústica cuyo tono irá aumentando, dejando se sonar cuando se haya cerrado el circuito eléctrico de la hebilla.

Figura 33. Alerta de cinturón

- **Caja Negra (EDR)**

  Tienen como fin recopilar información, tanto del vehículo como de sus ocupantes, registrando y almacenando los datos para que, en caso de accidente, poder conocer lo que ha ocurrido antes, durante y después del mismo.

  En el accidente, el EDR (Event Data Recorder o Registrador de Datos de Eventos) grabará los datos durante los 30 segundos previos y los cinco posteriores, el cual está situado normalmente bajo el asiento del conductor, atornillado al chasis como se puede ver en la siguiente imagen. Esta información incluye parámetros como la velocidad del vehículo, el frenado, el acelerador, el ángulo del volante, la activación de airbags, así como la fuerza del impacto en caso de colisión. Estos datos son muy importantes por investigadores de accidentes y autoridades para determinar causas y circunstancias de accidentes.

Figura 34. Caja negra o Módulo EDR

3. Metodología

Para la elaboración de la seta de emergencia de ahora en adelante SE, se basó en la anterior seta que se fabricó en el INSIA, Figura 35. La placa elaborada integraba una serie de funcionalidades simples pero efectivas, cortando la alimentación para liberar el sistema autónomo permitiendo al conductor actuar. Por tanto, para la mejora de esta placa se han listado las mejoraras necesarias a incluir en el nuevo sistema:

- Envío y recepción de mensajes CAN a la centralita
- Sistema de Leds que permitan conocer el estado de la placa
- Integración de un aviso sonoro cuando la seta está activada
- Montaje robusto y duradero

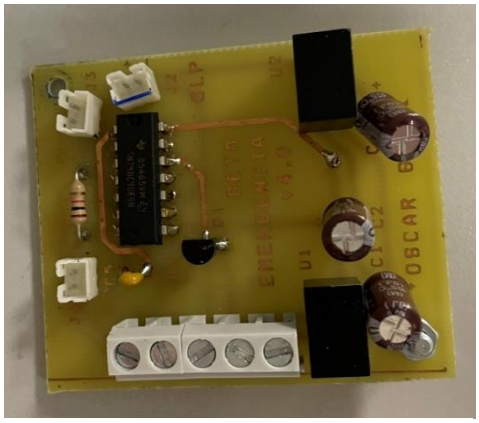


Figura 35. Antigua versión Seta de Emergencia

3.1. Hardware y Software

A continuación, se describirá el Hardware y el Software utilizado.

- **Hardware**

  - ***Arduino nano***: para controlar el sistema se ha utilizado un Arduino Nano, tras una investigación de elección de placa controladora se optó por la que tiene mayor facilidad de soporte y continuidad sin necesidad de requerir personal técnico especializado como es el caso de *peak*, ya que es un prototipo utilizado para un vehículo de investigación es muy susceptible a ser actualizado con frecuencia. Como características, es una placa compacta y versátil que ofrece potencia de procesamiento suficiente para ejecutar el software programado para el funcionamiento del sistema. Además, su pequeño tamaño hace posible su integración en aplicaciones donde el tamaño es limitado, como es el caso de este proyecto.

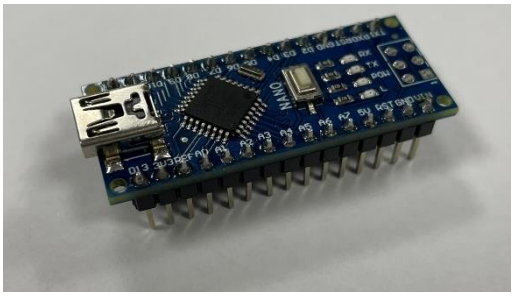


Figura 36. Arduino Nano

- ***Switches***: se han utilizado dos switches mostrados en la siguiente imagen, uno encargado de la activación del armado del sistema, el cual es el que integra el LED encargado de mostrar al usuario cuándo la SE está en modo armado, y el siguiente es el Switch principal que es pulsado cuando ocurre un suceso que no es esperado. Todo esto programado en una placa Arduino Nano.

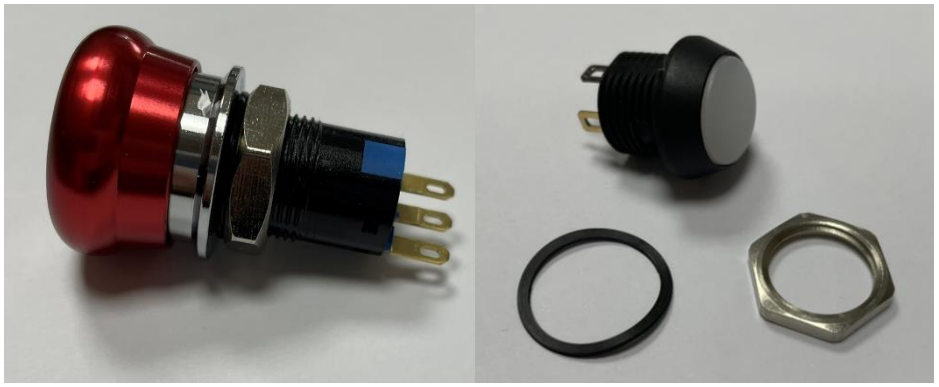


Figura 37. Switches

- ***MCP2515***: dispositivo encargado del envío y recepción de mensajes CAN en el que se conecta el cable CANLow y CANHigh en su lugar correspondiente.

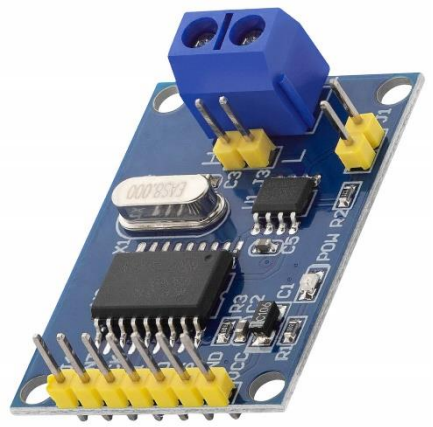


Figura 38. MCP2515

- ***Buzzer***: para el aviso sonoro del sistema se ha incorporado un Buzzer el cual se activa cuando se ha pulsado el switch principal. Para su correcto funcionamiento se ha incorporado un para regular la intensidad del sonido, un transistor NPN y las resistencias necesarias para que el circuito diseñado funcione correctamente.

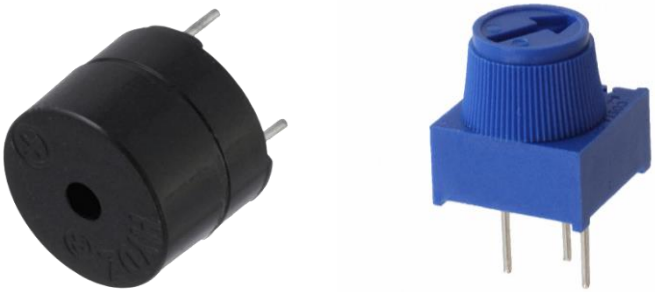


Figura 39. Buzzer y potenciómetro

- ***PCB***: para integrar todo este Hardware se ha diseñado e impreso en PCB una placa la cual incorpora todo el Hardware anteriormente descrito y además se han añadido 2 LEDs, uno para el encendido del sistema y el siguiente para controlar cuándo se está enviando la señal al buzzer. Para regular la tensión introducida externamente 12v, se ha diseñado un circuito regulador de tensión el cual corrige la tensión a 5v, cuyo condensador se puede apreciar, aunque este componente es propenso a calentarse ya que tiene que hacer una reducción de voltaje importante, se ha decidido no integrar un disipador externo, como se entrará en la sección 4.2.2 en la que se trabaja en este cálculo.

- ***Peak***: utilizado para la parte de la depuración del código. Este dispositivo permite la conexión entre la placa MCP2515 y un computador. De esta forma se ha podido visualizar los mensajes CAN que se envían en cada uno de los estados de la SE, pudiendo también hacer peticiones de mensajes CAN, que en general esperan una respuesta. Con este dispositivo se ha podido ver qué errores ocurrían en el sistema cambiando así las líneas de código correspondientes.

Figura 40. PCAN-USB

- **Software**

  En cuanto al software, el sistema ha sido programado en el lenguaje de programación C++ y desarrollado mediante el IDE Visual Studio, pero con algunas configuraciones.
  En lugar de utilizar la extensión de Arduino, o el propio programa que ofrece el mismo, se ha decido por la experiencia previa que se ha tenido con Arduino IDE y la necesidad de la incorporación de librerías externas para la gestión del MCP2515 de utilizar PlatformIO.

  PlatformIO es un IDE de código abierto para C/C++ [20], orientado al Hardware, ofreciendo varias herramientas para el desarrollo muy útiles en cuanto a la gestión del proyecto y librerías, en contra con Arduino IDE que no dispone de estas además de los problemas que presenta a la hora de abrir un código ya que no siempre utiliza la misma librería debido a actualizaciones que pueden crear incompatibilidades. PlatformIO está desarrollada sobre VSCode, ofreciendo varias ventajas:

  - Autocompletar código
  - CLI incluida (Arduino IDE no incluye)
  - Buscador y administrador de bibliotecas
  - Navegación por el código
  - Además de incluir varias ventajas que ofrece VSCode y sus posibles extensiones
  - Detección automática de puertos
  - Compilación y depuración del código más sencilla.

Para el control de versiones, se ha utilizado la plataforma Github, creando un repositorio propio para el proyecto, como se puede ver en el siguiente enlace enlace Github [1].
Una vez programado el código y para su depuración, se ha utilizado el programa PCANView, programa desarrollado por la empresa PEAK system. Este programa permite visualizar qué mensajes se están recibiendo del MCP2515, pudiendo enviar peticiones CAN para ver qué resultados arroja el sistema gracias al PCAN-USB explicado en el apartado anterior.

[1] Enlace GitHub: https://github.com/SIMCA-USI/Seta_Emergencia/tree/Version8

4. Desarrollo

En esta sección se detallará el proceso completo de la elaboración del sistema de Emergencia, pasando por el código implementado, funcionalidades del sistema, el proceso de diseño e impresión de la PCB, así como una explicación detallada del funcionamiento de la seta de emergencia.

4.1. Código Implementado y Funcionalidades del Sistema

Las funcionalidades del sistema son simples, al pulsar el botón principal, coloca el sistema en un estado denominado *Desarmado*. Además, existe un botón adicional que, al ser pulsado, coloca el sistema en un estado *Armado*, y cuando se vuelve a presionar el botón principal, el sistema vuelve a *Desarmado*. Los botones correspondientes a esta función son los mostrados en la Figura 37:

El código también gestiona la activación de un LED y un buzzer dependiendo del estado del sistema, y es capaz de enviar mensajes CAN para informar sobre el estado actual. Como ejemplo se tiene el mensaje CAN siguiente:

CAN-ID	Type	Length	Data
602h		8	00 00 00 00 00 00 00 02

Figura 41. Ejemplo de mensaje CAN

Aunque se entrará en más detalle en la siguiente sección, la figura muestra un mensaje can de una seta de emergencia situándose en el estado *Armado*.

A continuación, se explicará cada sección del código de forma detallada.

4.1.1. Void y funciones auxiliares

La función *void()* es la encargada de la inicialización de las variables además de la asignación de las entradas y salidas que dispone el Arduino. Por tanto se empieza por inicializar la comunicación serial a una velocidad de 9600 baudios, el mcp2515 se establece a una velocidad de transmisión de 500Kbps con un oscilador de 8Mhz y se añaden las líneas necesarias para que pueda recibir y enviar mensajes. Se configuran los pines de los switches de Seta de Emergencia y el botón Led con entradas de resistencias Pull-Up, es decir, se leerá un valor High a menos que sean pulsados que se leerá un valor Low, finalmente se configuran los pines del Led y el buzzer como salidas.

Para que se puedan utilizar interrupciones en el programa se debe de usar la función attachInterrupt con el pin que se quiera utilizar la interrupción, en este caso se han utilizado los dos switches de la seta para que sean utilizados como interrupciones. Para lograr esto se deben de tener pines en la placa de desarrollo que acepten estas interrupciones, en este caso, el Arduino Nano dispone de dos pines que soportan la función de interrupción y son los pines digitales 2 y 3.

Se han programado dos funciones auxiliares las cuales se llaman en la parte del loop como se va a explicar más adelante. La primera función programada es la que se encarga de las interrupciones del sistema, como se tienen dos interrupciones se ha programado dos, una para cada interrupción. Estas funciones tienen una función básica que es la de cambiar una variable a true cuando es pulsado uno u otro switch, así se es capaz de identificar la interrupción y pasar al código correspondiente.
La función *attachInterrupt* necesita que le pasen 3 parámetros, el pin en el que actúa, la función a la que llama (*callback*) y cuándo se quiere que se active la función, en este caso cuando está a nivel

Low se quiere que se active, por eso se indica *Falling* en este parámetro. Si se quisiera activar cuando el pin detecte un High se debería de poner *Raising*. Y si se quiere que se active indiferentemente si se detecta un High o Low se debe poner *Changing.*

```
attachInterrupt(digitalPinToInterrupt(pinSetaEmergencia), interrupcionEmergencia, FALLING);
attachInterrupt(digitalPinToInterrupt(pinBotonNormal), interrupcionBotonNormal, FALLING);
```

Figura 42. Interrupciones

La última función auxiliar programada es la encargada de seleccionar el número de seta, leyendo los pines analógicos del Arduino del 0 al 3. Así se tienen 4 variables las cuales nos proporcionan cuál switch se ha levantado para simular qué seta se ha seleccionado. Como se tienen 4 pines se pueden seleccionar hasta 16 setas y simplemente esta función va sumando 1, 2, 4 u 8 según si se ha pulsado el pin 0, 1, 2 o 3 respectivamente.

4.1.2. Loop

Esta función es la encarga de contener las acciones que va a realizar el Arduino de forma continua, primeramente se declaran las variables locales para cada acción a realizar y seguidamente se añaden el código correspondiente.

- Declaración de dos variables locales, *estadoSetaEmergencia* y *estadoBotonNormal*, se les asigna el valor de lectura de los pines *pinSetaEmergencia* y *pinBotonNormal*, que son respectivamente el pin digital 2 y 3. Estos pines se utilizan para leer el estado de los botones o conectados.

- Se declara una variable *currentTime* y se le asigna el valor actual del tiempo en milisegundos utilizando la función *millis()*. Esta función devuelve el tiempo transcurrido desde que se encendió el sistema.

- Se verifica si ha pasado al menos un segundo desde la última vez que se ejecutó la función *getDIPValue()* y si el estado es igual a *DesarmadoFinal*. Si ambas condiciones son verdaderas, se llama a la función getDIPValue() que es la encargada de seleccionar el número de seta gracias a unos microSwitches integrados en la placa, y se asigna el resultado a la variable *númeroSeta*. Además, se actualiza la variable *lastTime* con el valor actual de *currentTime*.

```
if (currentTime - lastTime >= 1000 && estado == DESARMADO2) {
    numeroSeta = getDIPvalue();
  lastTime = currentTime;
}
```

Figura 43. Selección de número de seta

- Se configuran los valores de la estructura *stmp* que se utiliza para enviar mensajes CAN. Los valores de los campos *can_id*, *candlc* y *data* se establecen según la seta seleccionada y los estados del sistema.

- El mensaje CAN enviado tiene un payload que sigue el estándar CANOpen. Se ha decidido enviar por el ***can_id*** número 602 ya que es el que está libre en el vehículo de pruebas del INSIA. El can_dlc indica cuántos bytes se van a transmitir en este caso8 bytes. Seguidamente se envían los datos del sistema que se divide en 4 secciones:

- **Specifier**: en que se indica el propósito del mensaje. Si este campo contiene **20** quiere decir que se ha enviado un mensaje CAN que espera una respuesta, y es la respuesta en el que en este campo debe contener un **40** que indica que el mensaje que se está enviando es una respuesta. Si el campo contiene **00** quiere decir que se está enviando el mensaje de forma normal y que no ha sido generado de un mensaje CAN que espera respuesta.

- **Index** - **Subindex:** estos campos no son utilizados en este sistema, por lo que se rellenan con 0.

- **Data**: este campo es el que se utiliza para enviar información importante sobre el sistema. En este caso se utilizan los 2 últimos bytes en los que el byte[6]

```
stmp.can_id=0x602;
stmp.can_dlc=8;
stmp.data[0]=0x00;
stmp.data[1]=0x00;
stmp.data[2]=0x00;
stmp.data[3]=0x00;
stmp.data[4]=0x00;
stmp.data[5]=0x00;
stmp.data[6]=numeroSeta;
stmp.data[7]=0x00;
```

Figura 44. Inicialización mensaje CAN

indica el número de seta seleccionada y el byte[7] indica el estado actual del sistema siendo **0x01 Desarmado¸0x02 Armado**, **0x03 Pulsado**.

En la siguiente figura se mostrará un mensaje CAN del sistema:

CAN-ID	Type	Length	Data
602h		8	00 00 00 00 00 00 03 02

Figura 45. Ejemplo mensaje CAN

Como se puede observar, el mensaje tiene el identificador CAN 602 en hexadecimal, la longitud de datos son 8 bytes. En el Byte 0 del campo de datos indica que el mensaje no ha sido producto de una petición del usuario El byte 6 proporciona el número de seta que se está seleccionando, en este caso la seta número 3. El byte 7 proporciona el estado de la seta número 3, en este caso el 02, que se corresponde con el estado *Armado*.

- Se utiliza una estructura de control **switch** para que realice diferentes acciones según el valor de la variable *estado* ejecutando una parte u otra según el estado en el que se encuentre el sistema enviando un mensaje CAN según el estado del switch.

- En el caso de **Desarmado**, se verifica si el estado de la seta de emergencia, *estadoSetaEmergencia*, es de nivel bajo *LOW*. En caso de ser así, se transita al estado **Armado**, se envía el mensaje CAN utilizando la función *sendMessage()* cada segundo además *de* realizar una serie de acciones, como imprimir un mensaje en el puerto serie, cambiar el estado de un

LED y agregar retrasos (delays) para que el sistema no se quede trabado.

Seguidamente, se verifica si se ha recibido un mensaje CAN con identificador 0x20 utilizando la función *readMessage()*, y se procede a responder como se ha explicado en los apartados anteriores.

- En el estado **Armado**, se enciende un LED y se configuran los valores del mensaje CAN y se verifica el estado del sistema de emergencia. Si está alto (HIGH), se envía el mensaje CAN correspondiente y se envía cada 5 segundos, parte que se corresponde con el bucle while.
  Si en algún momento de este bucle se pulsa la seta de emergencia inmediatamente se sale del bucle y se transita al estado **Pulsado** y si se recibe un mensaje CAN de petición de respuesta se envía la respuesta correspondiente con el formato antes mencionado.

- En el caso de **Pulsado**, se configuran los valores de la estructura *stmp* y se envía el mensaje CAN cada 50ms siguiendo las directrices de lo explicado anteriormente utilizando la función *sendMessage()*. Seguidamente, se controla el estado del buzzer y del LED, alternando su estado cada segundo utilizando la función *digitalWrite()* y la variable *buzzerActive* .
  A continuación, se verifica si el estado del switch de la seta de Emergencia es alto (LOW). Si es así, significa que el switch de la seta de emergencia se ha soltado y finalmente se transita al estado a **Desarmado**.

4.2. Diseño e Impresión de la PCB

En esta sección se explicará el procedimiento que se ha realizado en cuanto al diseño e impresión de la placa.

4.2.1. Programas utilizados

Se han utilizado el programa Kicad versión 7.0 para los circuitos diseñados. Kicad es un paquete de software libre para la automatización del diseño electrónica **EDA** (Automatización de diseño Electrónico) utilizado para el diseño de **PCB** (placas de circuito impreso o circuitos electrónicos). Es un software perfecto por su simplicidad y facilidad de uso, permite realizar una amplia variedad de tareas relacionadas con el diseño de circuitos electrónicos y fabricación de las placas. Su gama de funcionalidades cubre casi todas las etapas de diseño de PCB, desde la creación de esquemáticos hasta la generación de archivos de fabricación. Además de disponer de editor de esquemas, huellas y visor 3D [21].

Entre otras funcionalidades, Kicad te permite crear tus propias bibliotecas, muy interesante si quieres hacer añadir un componente que no está en las bases de datos genérica del programa.
Se ha decidido utilizar este programa por diversos motivos entre los que se encuentran el

conocimiento previo de la utilización de Kicad en la asignatura Sistemas Empotrados, código abierto y gratuito, completo conjunto de herramientas, comunidad activa, herramientas de software multiplataforma y no hay necesidad de licencias. Cabe destacar que Kicad es un programa de continuo desarrollo y puede contener errores, por lo que la comunidad agradece mucho si se comunica un error en la utilización de alguna herramienta a través de alguna comunidad como GitHub o Reddit.

A parte del programa de diseño, se ha utilizado la página principal de JLCPB. Para el diseño de alto nivel se ha utilizado el programa Papyrus, el cual ya ha sido utilizado en la asignatura de *Sistemas Empotrados*. Este programa ha sido útil para la creación de un diseño el cual muestra el trabajo realizado tanto de Hardware como del Software.

4.2.2. Diseño de la placa

El primer paso que se dio en Kicad fue añadir todos los componentes en el esquemático. Como el Arduino Nano no se dispone de forma predeterminada, se procedió a añadirlo manualmente. Cuando ya se tenía claro en qué pin va a ir cada componente se procedió a añadirlo al esquemático con una etiqueta, esta se conectará en el pin que corresponda, como ejemplo se muestra la Figura 46.

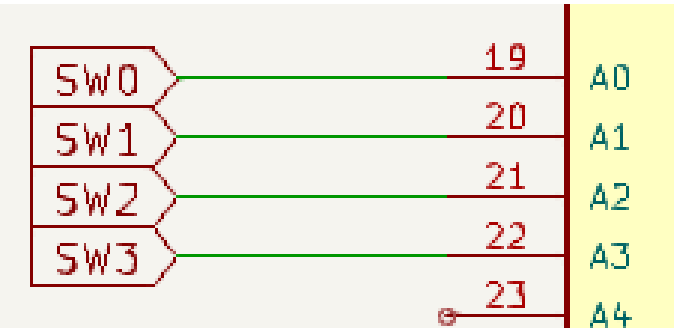


Figura 46. Asignación de etiquetas del switch

Seguidamente se añadieron las demás huellas a utilizar, es decir, el Dip Swtich, el inversor de voltaje y la placa MCP2515.
Se divide la hoja en diferentes secciones en este caso 6 secciones una para cada uno de los componentes que incorpora el sistema:

- **Circuito selectores de setas**
  Se ha incorporado un Dip Switch el cual contiene 4 micro switches, cada uno de ellos necesita una resistencia de Pull-Up como se ha mencionado anteriormente. La función de esta resistencias es la de "empujar" hacia arriba lo que se denomina como polarizar el voltaje hacia el voltaje de fuente (VDD) en este caso 3.3V [22]. De esta forma cuando el pulsador está abierto o en reposo, el volataje en el pin del Arduino será de 3.3V. Es importante resaltar que las entradas de Arduino son de alta impedancia, es decir que la corriente que circulará por esa línea será mínima en el orden de los micro-amperios, por lo que el voltaje que "cae" en la resistencia de Pull-Up es mínimo y tendremos el mismo voltaje de fuente en la entrada del Arduino.
  Cuando el pulsador se presiona, la corriente circula por la resistencia y seguidamente por el pulsador, así se tendrá el voltaje de Tierra (0v o Low) en la entrada del Arduino.

  Cuando ya se tiene diseñado el circuito se deberán añadir las etiquetas que indicarán en qué pin del Arduino se conectan. En este caso se ha añadido la etiqueta de 3.3V que irá al pin correspondiente del Arduino de 3.3V, la etiqueta GND la cual va a una de los dos pines de GND, y seguidamente las 4 eqitques de los switches que se conectarán a los pines analógicos del 0 al 3. En la siguiente imagen se puede observar el circuito que se ha diseñado con los símbolos que se han explicado en este párrafo.

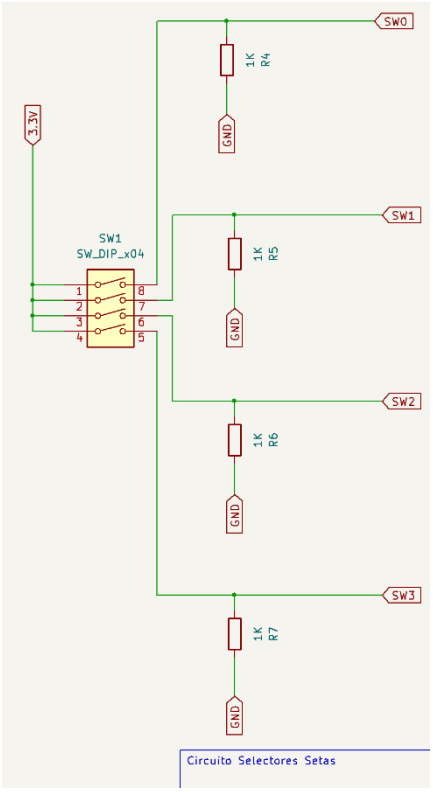


Figura 47. Circuito Selectores Setas

- **Circuito Seta y BotonLED**

  En esta sección se ha diseñado un circuito simple, ya que se han simulado los dos switces, Boton seta de emergencia y Boton LED. Se dispone de la etiqueta de 3.3V y a otras dos nuevas que indicarán en qué pin del Arduino se conectarán. En este caso el pin de seta de Emergencia y BotonLED van conectadas al pin digital 2 y 3 respectivamente como se puede ver en la siguiente Figura 48.

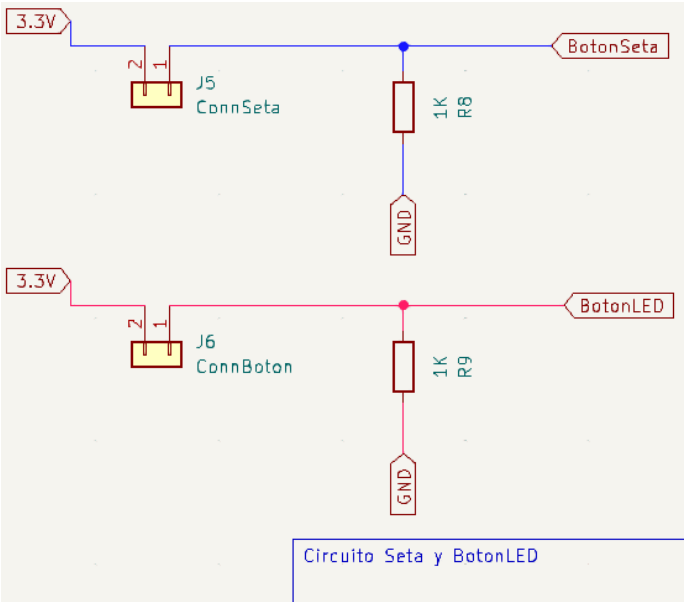


Figura 48. Circuito Seta y BotonLED

- **Circuito Buzzer**

  El buzzer está conectado a una alimentación auxiliar de 12V, ya que es la alimentación que proporciona un vehículo por norma general. Es por eso que se ha decidido usar un buzzer activo. Estos buzzers se caracterizan por que generan un sonido de una frecuencia determinada y fija cuando son conectados a tensión, incorpora un oscilador simple por lo que únicamente es necesario suministrar corriente al dispositivo para que emita sonido [23]. Entre las ventajas que presenta un buzzer activo se encuentra la de no suponer carga para el procesador.

  Por el contrario un buzzer pasivo necesita que le proporcionen una señal eléctrica para convertirla en una onda de sonido. Estos dispositivos no disponen de electrónica interna, por lo que se debe proporcionar la onda desde el controlador. Pudiendo variar el tono emitido modificando la señal que aplicamos al altavoz, lo que permite generar melodías [24], en el que una onda más larga generaría un tono más grave, y una onda más corta un tono más agudo.

  A parte del buzzer se ha añadido un transistor PnP el cual tiene como función dejar la señal procediente del Arduino para generar ondas eléctricas simulando un PWM, para que el circuito tenga la capacidad de añadir un buzzer pasivo en caso de necesidad.

  De esta forma el transistor funcionaría en sincronía con el potenciómetro suministrando los 12V al buzzer y dejaría pasar la señal de onda para que el buzzer tenga la capacidad de generar un sonido agudo cuando la longitud tenga valores pequeños y un sonido grave cuando la longitud de onda tenga valores más altos. Todo el circuito necesita de resistencias y también de las etiquetas de las cuales únicamente la de señal (SIG) es conectada Arduino, concretamente en el Pin digital 5.

  Por último, se ha diseñado un LED, el cual tiene como objetivo encenderse cuando se envíe una señal a través de la etiqueta SIG. Esto permite saber al usuario cuándo se está enviando una señal de activación del buzzer y por tanto resulta útil para hacer

pruebascuando no se quiere que el buzzer emita sonido. A continuación se muestra la imagen del circuito.

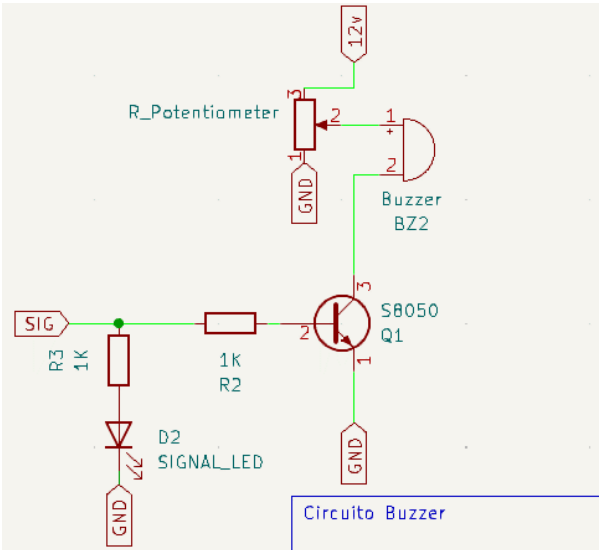


Figura 49. Circuito Buzzer

- **Circuito Regulador de Tensión**

  Este circuito tiene como función reducir la tensión de la entrada de 12V ya que es la que normalmente proporcionan la mayoría de los vehículos, es la que alimentará el sistema a 5V estables, por eso se tienen dos condensadores y un terminal de regulador de voltaje. Para saber si es necesario un disipador de calor se ha creado una tabla [25] la cual introduciendo los valores de las temperaturas máximas del ambiente , las de las uniones de los componentes y volaje que transforma el sistema, se calcula la potencia máxima disipable y la potencia disipada por el sistema. Con estos valores se puede conocer si es necesario un disipador si el valor de la potencia disipada por el sistema es mayor que la potencia disipable. En este caso como se puede ver, el resultado es que no es necesario un disipador de calor extra.

Estudio de potencia disipada y calor generado							
Study of disipated power and generated heat							
	Temp. Union P-N	Tj	125				
	Temp. Max. Ambiente	Ta	30		JC	CA	
		Θja	23,9		23,9	16,7	
	Potencia máxima Disipable (W)	3,9748954					
	Voltaje de entrada	Vin	12				
	Voltaje de salida	Vout	5		Potencia 5v	Potencia 3v	Potencia
	Corriente consumida	I	119		570	16,5	586,5
	Potencia Disipada por el sistema	0,833					
	¿Es necesario incluir disipador térmico?			NO			
	¿It is mandatory to include a thermal disipator?						

Tabla 1. Disipación de Energía

En el circuito también se ha diseñado un Led el cual se enciende siempre cuando el sistema está encendido, teniendo como función un sistema extra de seguridad en el sistema, como se observa en la Figura 50.

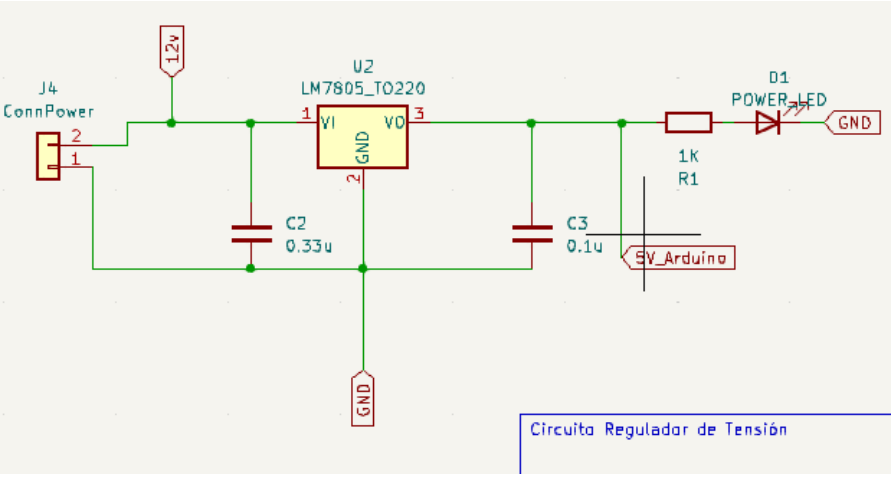


Figura 50. Circuito Regulador de Tensión

- **Circuito MCP2515**

Para que sea posible el envío y recepción de mensajes CAN en el Arduino Nano, es necesario añadir un módulo externo, en este caso se ha añadido el módulo MCP2515 útil para conectar sistemas y dispositivos a un bus CAN utilizando la interfaz SPI para poder comunicar microcontroladores.

El circuito contiene el módulo mencionado, un connector JST de 2 pines el cual proporciona el CAN High y el CAN Low, y las etiquetas de alimentación, tierra y los módulos *CS, S0, S1, SCK, INT* que necesita el módulo para su correcto funcionamiento observándose en la Figura 51.

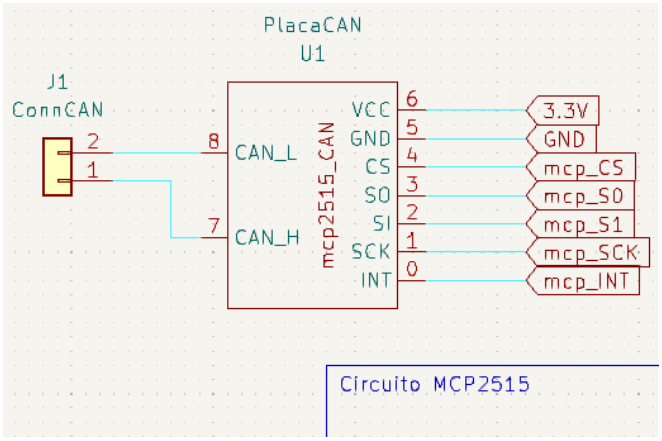


Figura 51. Circuito MCP2515

- Conector Auxiliar: Este conector de 4 entradas mostrado en la Figura 52. Conector Auxiliar, simplemente está conectado a dos pines Analógicos A6, A7 y dos pines Digitales D0 y D1. Está diseñado así por si en algún futuro se quiere añadir alguna funcionalidad extra y así facilitar la instalación de los nuevos componentes sin tener que alterar el diseño de la placa.

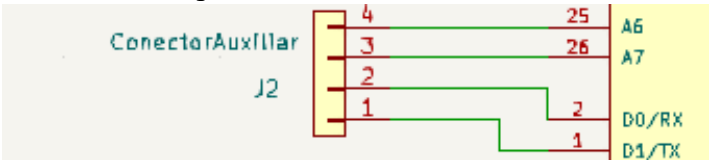


Figura 52. Conector Auxiliar

- **Placa Arduino**

  Por último, se recopilarán las entradas utilizadas para cada uno de los circuitos explicados en los apartados anteriores, donde las abreviaturas son A*X* (Analógico), D*X* (Digital) siendo *X* un número del 0 al 7 e *Y* un número del 0 al 13.

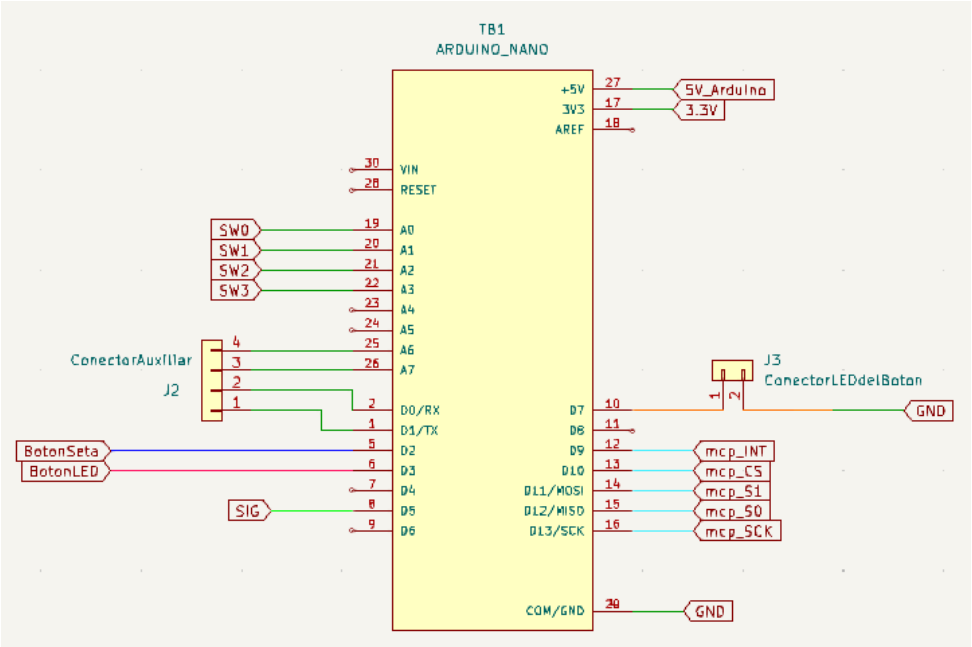


Figura 53. Arduino y Pines utilizados

Una vez se tienen diseñados cada un de los circuitos del sistema, se procede a añadir las huellas de cada uno de los componentes para poder diseñar la PCB, se deben de seleccionar una a una las huellas de cada uno de los componentes, resistencias, LEDs, condensadores, Arduino, MCP2515, etc.

Para poder importar las huellas al espacio de trabajo de la PCB se debe de ir a la ventana de Herramientas y pulsar Actualizar placa desde el esquema. Automáticamente se añadirán todos las huellas asignadas en el esquemático al espacio de trabajo. Una vez importadas, se procede a diseñar la placa añadiendo la forma que se desea que tenga. En este caso se han tomado las medidas de la estructura metálica que se tiene para que la placa encaje en ella, por eso se procede a añadir un rectángulo con las medidas tomadas en la capa *Edge.Cuts*. En esta capa se diseñan todos los crotes que se quiere que haga la empresa encargada de la fabricación de la PCB, en la Figura 54 se puede ver los componentes importados y la placa diseñada.

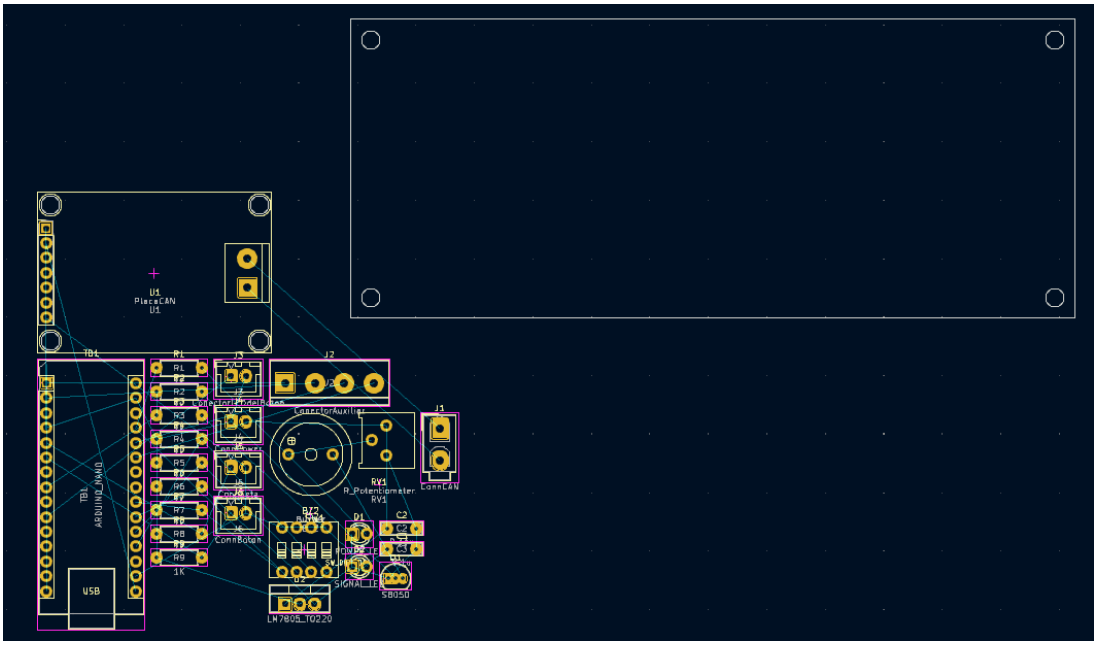


Figura 54. Componentes y placa

Dentro del rectángulo se proceden a encajar todos los componentes del sistema en la Figura 55 se muestra el resultado final. Una ayuda que presenta Kicad es que muestra qué componentes deben de estar interconectados, muy útil ya que con esta ayuda los componentes que tienen alguna relación deben de estar próximos entre sí para que las pistas no “ensucien” el espacio de trabajo.

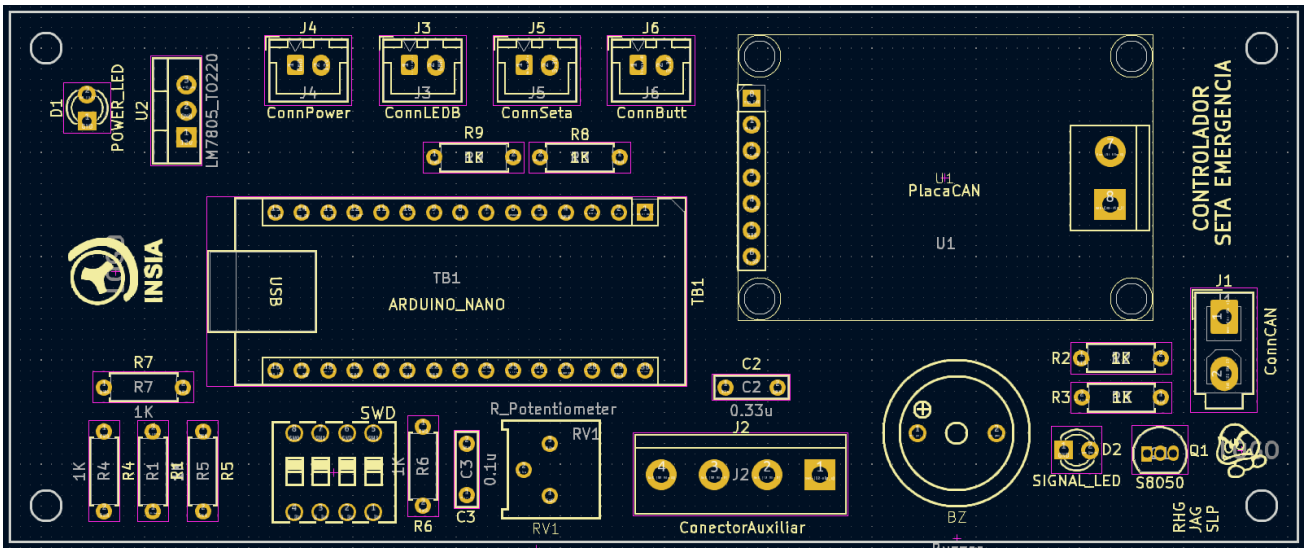


Figura 55. Diseño de la placa final

Una vez los componentes están ajustados, se procede a conectarlos con las pistas, estas se encuentran en *F.Cu* (Fron) y *B.Cu* (Back). Como se puede intuir, se pueden colocar las pistas en la cara frontal de la PCB o en la cara trasera, esto permite que haya más espacio para las pistas ya que estas no se pueden superponer, esto presentaría un corto circuito ya que estas pistas están en un espacio en dos dimensiones, por tanto no se pueden superponer. Kicad distingue estas pistas con colores distintos, rojo y azul respectivamente por defecto. De esta forma el ususario puede diseñar las pistas diferenciando con claridad en qué lugar de la placa están situadas.

Por tanto se procede a conectar los componentes con los pines de los componentes correspondientes con la ayuda de las líneas de guía que muestra Kicad como se puede ver en la Figura 57. Para una más cómoda y sencilla forma de conectar los componentes con las pistas se ha utilizado un programa llamado ***Freerouting***. Este programa genera las pistas necesarias de forma automática, conectando con la pistas de una manera eficiente y sin dificultad, como se puede ver en la Figura 58.

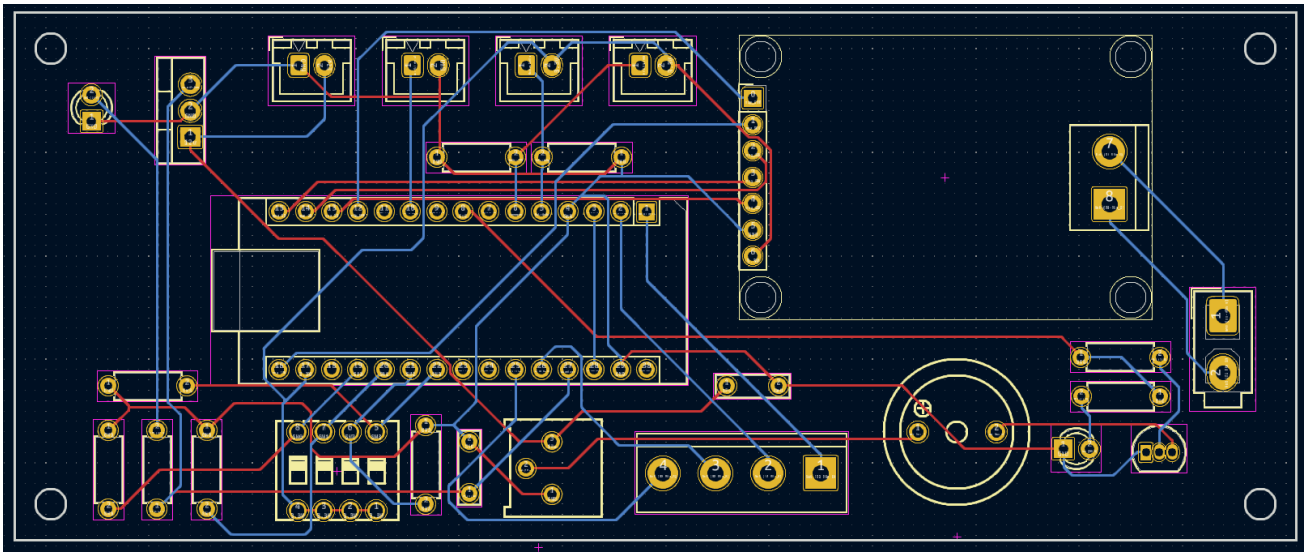


Figura 57. PCB con los componentes y pistas integrados

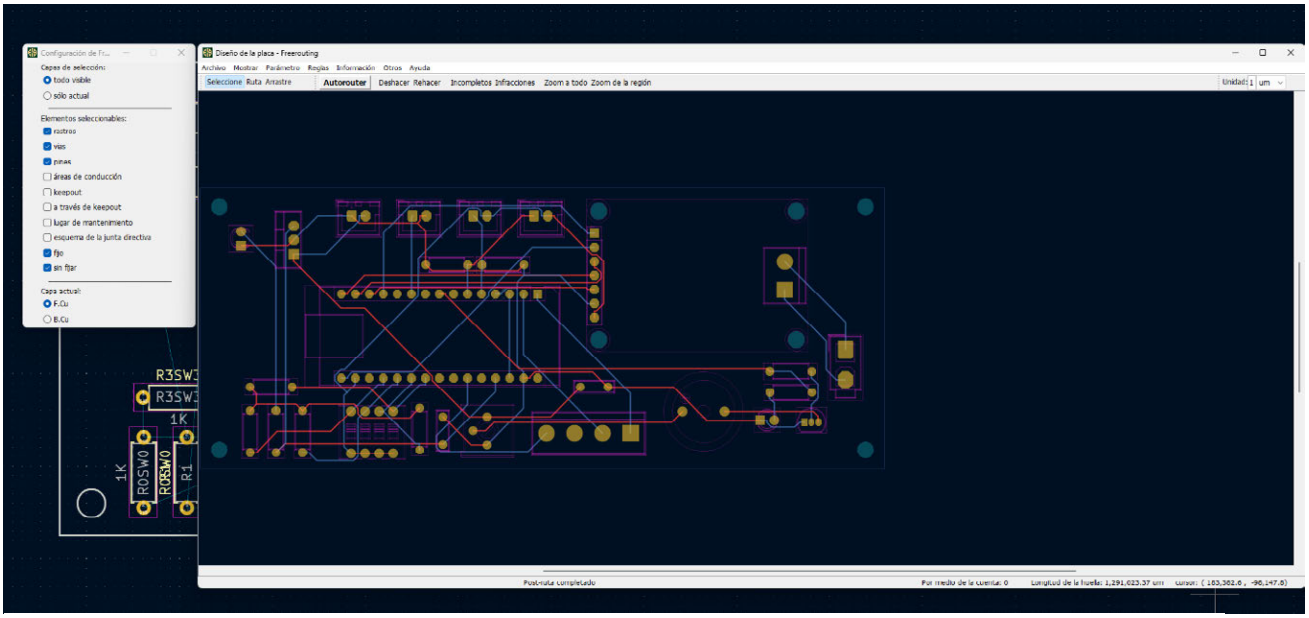


Figura 56. Conexión con Pistas usando el Software Freerouting

Es normal que los componentes se tengan que mover, ya que es complicado que se tenga un diseño final de estos desde el principio debido a que las pistas pueden impedir algunas conexiones. Una ayuda que se debe de tener en cuenta es que si se pulsa cualquier componente o pin en el espacio de trabajo en el esquemático se resaltará, esto es muy útil si se quiere averiguar a qué debe de estar conectado o a qué hace referencia ese componente o pin.

Una vez los componentes están situados correctamente, se proceden a introducir los textos que distinguen los componentes en la placa final. En este diseño de han comentado cada uno de los componentes, distinguiendo cada uno por su letra inicial. Como ejemplo, se tiene una resistencia, por tanto la nomenclatura sería R1, así con las demás resistencias R2, R3, etc. Se procede a hacer lo mso con los LEDs, Condensadores y los JSTs. Para el resto de los componentes se procede a poner el texto de la placa que se integra (ARDUINO_NANO, PlacaCAN, SWD para los microswitches y los conectores).

Por tanto el resultado final es que se muestra en la siguiente Figura 58.

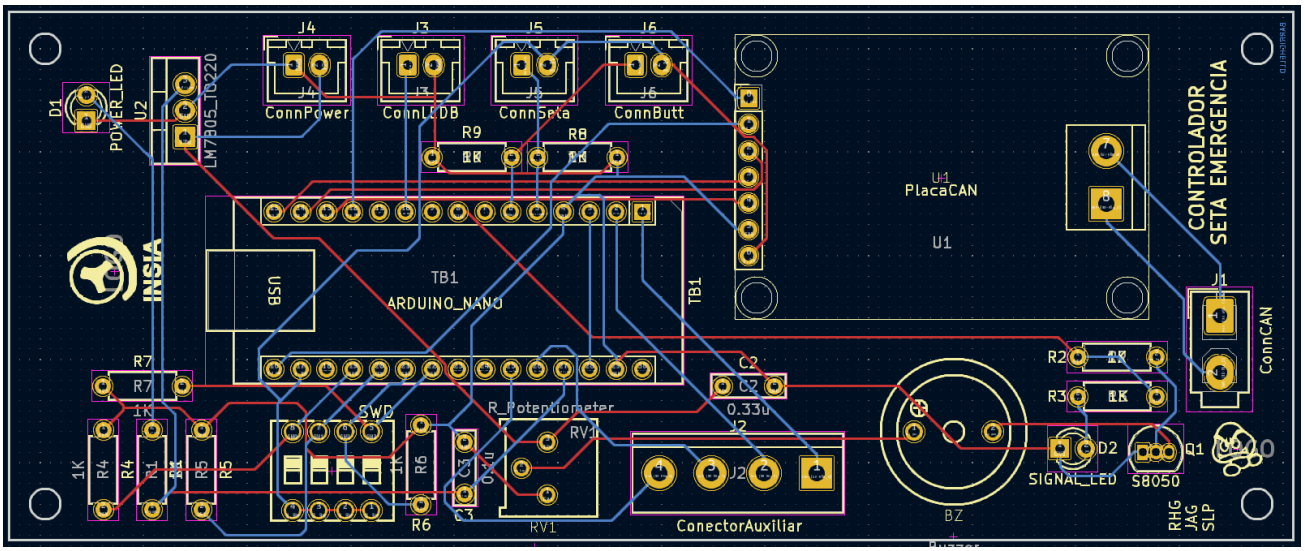


Figura 58. PCB Diseño Final

4.2.3. Impresión de la placa

Para la impresión se decidió hacerlo con la compañía ***JLCPCB***, ya que el INSIA ya había fabricado placas con esa compañía con un resultado muy bueno y un precio razonable. Para que JLCPCB pueda imprimir la placa y soldar los componentes, es necesario realizar unos pasos en kicad y en la página principal de la compañía, las instrucciones se pueden encontrar en este enlace [2].

- **Generación de archivos**

  Se deben de generar los archivos necesarios para la impresión, estos son los denomiados *Gerbers*. Para eso se debe de dirigir a Kicad esquemáticos en la ventana *Archivo* y clicar en *Archivos de fabricación*, y se verá la sección *gerbers(.gbr)*. Se recomienda tener el proyecto Kicad en una ruta en el equipo predefinida y tener varias carpetas para la correcta identificación de los archivos.

  Se seleccionan las capas que se ve en la Figura 59.
  Seguidamente, se procede a generar los *Drill Files*, en la misma ventana de los *Gerbers* se encuentra un botón que pone *Generar Archivos de Taladro*, indicado también Figura 59.

[2] Instrucciones de JLCPCB: https://jlcpcb.com/help/article/362-how-to-generate-gerber-and-drill-files-in-kicad-7

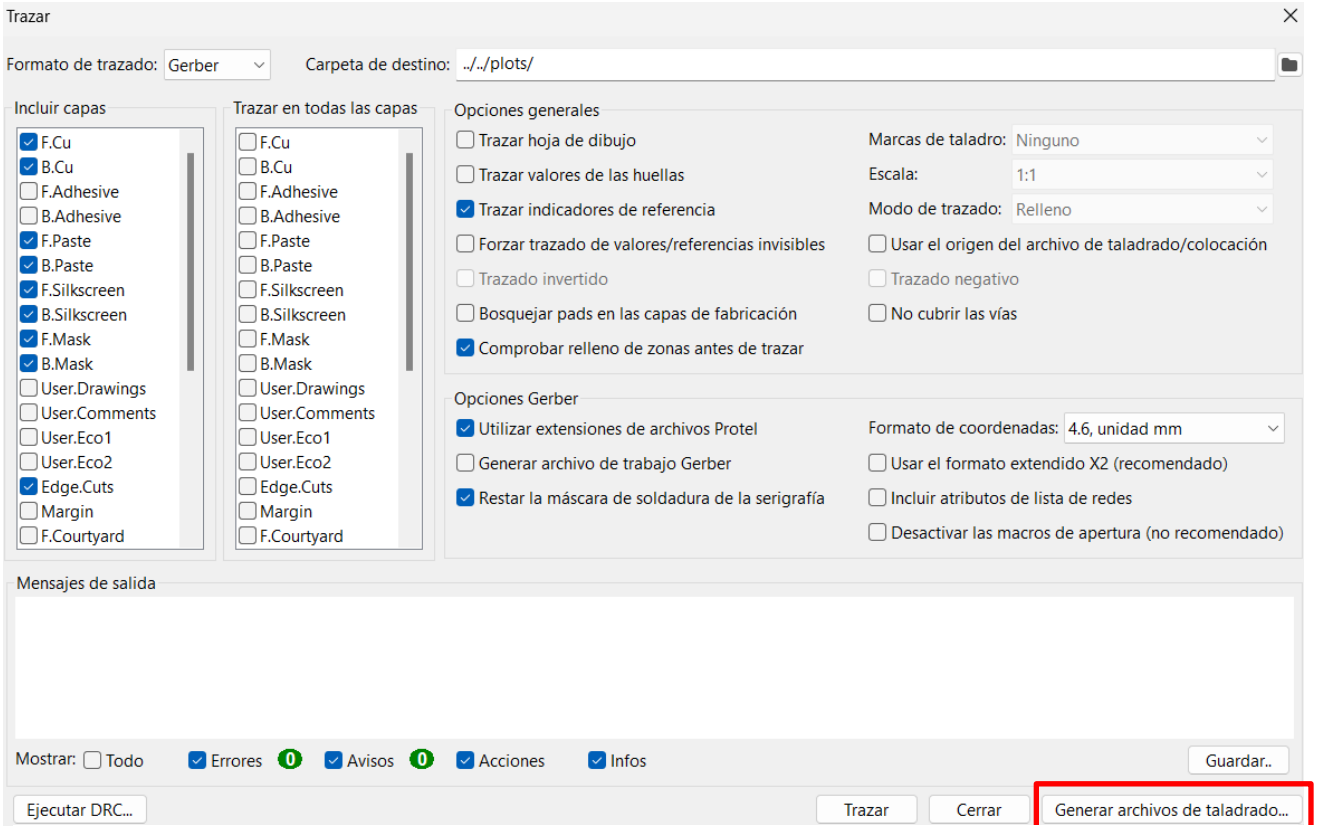


Figura 59. Archivos Gerbersl y Taladro

Una vez se tiene generados los *Gerberls* y los *Drill Files*, se debe de generar el *BOM*, estos simplemente son una lista de materiales para que JLCPCB sepa qué componentes y en qué posición deben de estar instalados. En la página de instrucciones antes enlazada en el documento no se muestra exactamente cómo generar el BOM de forma automática ya que se enseña a hacerlo para versiones más antiguas de Kicad, sin embargo, en las nuevas versiones de Kicad no funciona con las instrucciones dadas, sino que se hacen automáticamente con una extensión que se puede encontrar en este enlace de GitHub [3]. De esta forma se obtiene el BOM el cual está hecho en Excel y es un *.csv* como se puede ver en la siguiente Figura 60.

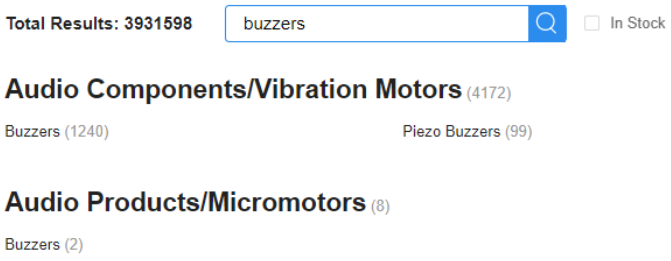
[3] Extensión para generar BOM: https://gist.github.com/gcormier/61addd28226bfd4ed6f645f312c3f13a

	A	B	C	D	E	F	G	H
1	Designator,Footprint,Quantity,Value,LCSC Part #							
2	BZ,BuzzerRedondo,1,Buzzer,C252953							
3	C2,C_Rect_L7.0mm_W2.0mm_P5.00mm,1,0.33u,C2831918							
4	C3,C_Rect_L7.0mm_W2.0mm_P5.00mm,1,0.1u,C2761729							
5	D1,LED_D3.0mm,1,POWER_LED,C99771							
6	D2,LED_D3.0mm,1,SIGNAL_LED,C99771							
7	J2,TerminalBlock_bornier-4_P5.08mm,1,ConectorAuxiliar,C430624							
8	J3,JST_XH_B2B-XH-A_1x02_P2.50mm_Vertical,1,ConnLEDB,C265283							
9	J4,JST_XH_B2B-XH-A_1x02_P2.50mm_Vertical,1,ConnPower,C265283							
10	J5,JST_XH_B2B-XH-A_1x02_P2.50mm_Vertical,1,ConnSeta,C265283							
11	J6,JST_XH_B2B-XH-A_1x02_P2.50mm_Vertical,1,ConnButt,C265283							
12	Q1,TO-92_Inline,1,S8050,C150546							
13	R1, R2, R3, R4, R5, R6, R7, R8, R9,R_Axial_DIN0207_L6.3mm_D2.5mm_P7.62mm_Horizontal,9,1K,C713997							
14	RV1,Potenciometro,1,R_Potentiometer,C116305							
15	SWD,DIP_4P_THT_LCSC,1,SW_DIP_x04,C99418							
16	U2,TO-220-3_Vertical,1,LM7805_TO220,C50931							
17								

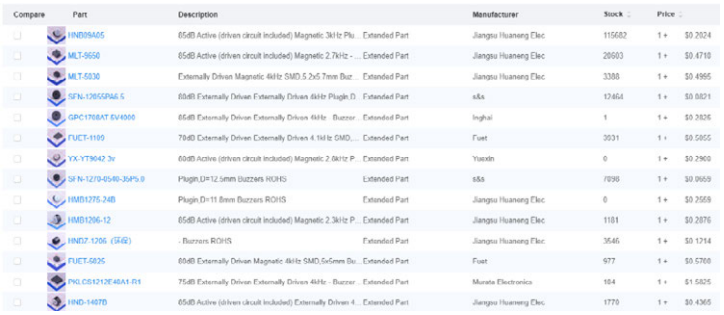
Figura 60. BOM

- **Subida de archivos a la página Web de JLCPCB y selección de componentes**
  Una vez se tienen los archivos de impresión necesarios, se deben de subir a la página web. Seguidamente se deben de seleccionar los componentes que se quiere integrar en la placa. Como ya se tienen las huellas en Kicad, se deben de buscar estas mismas en la web de JLCPCB, el enlace es el siguiente [4]. Como ejemplo se procede a explicar la búsqueda del buzzer de la placa.

1. Se busca el componente en la página web

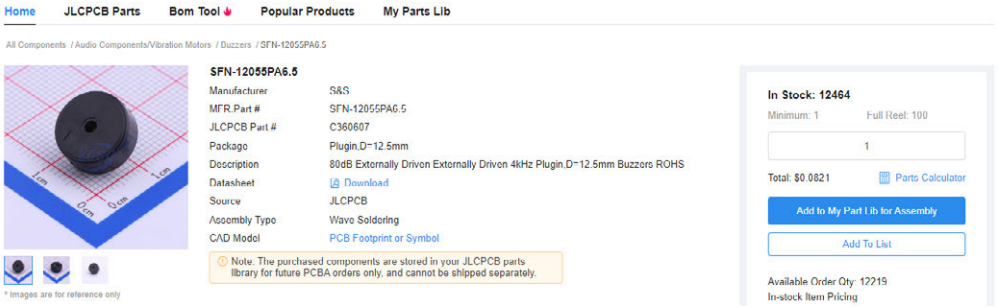


2. Se selecciona el buzzer con la referencia



[4] Selección de Componentes: https://jlcpcb.com/parts

3. Añadirla a la lista de montaje



Se procede a hacer lo mismo con todos los componentes de la placa. Los componentes utilizados en este proyecto son los indicados en la siguiente Figura 61.

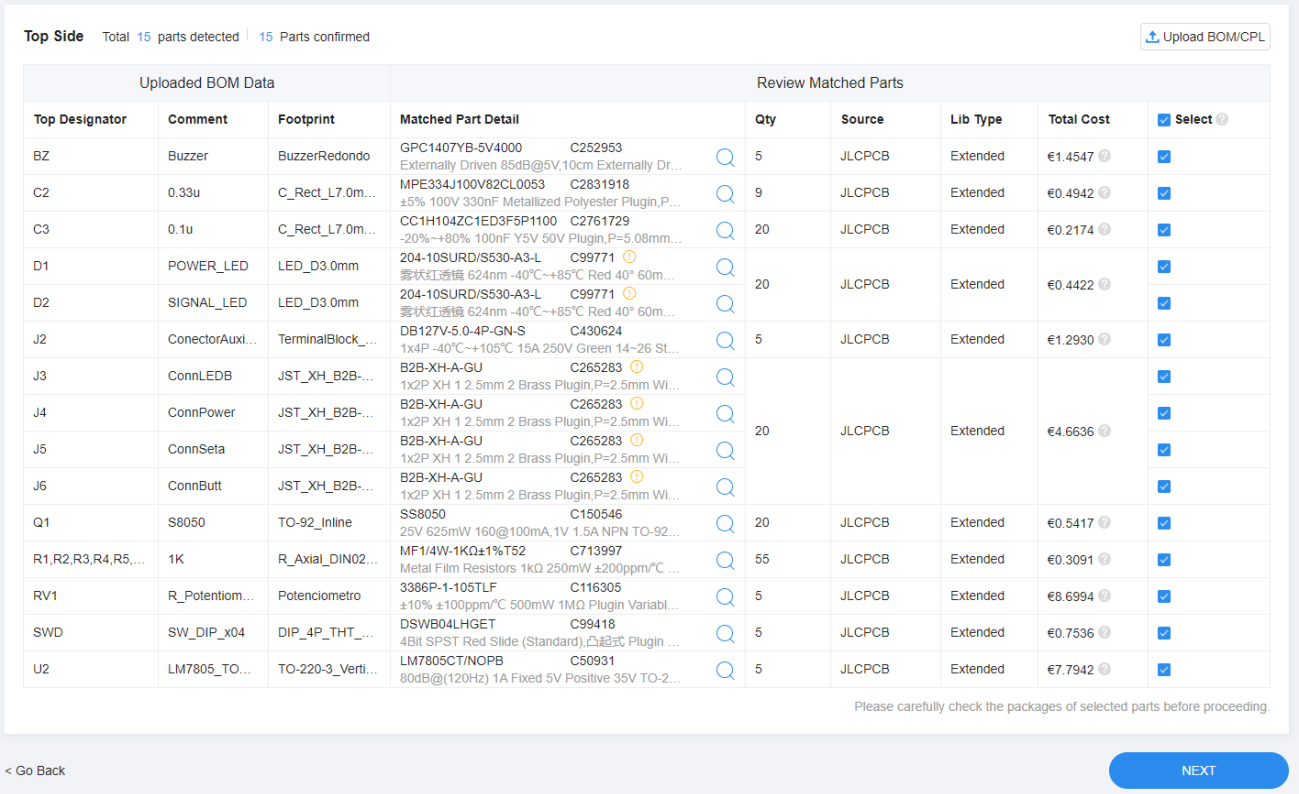
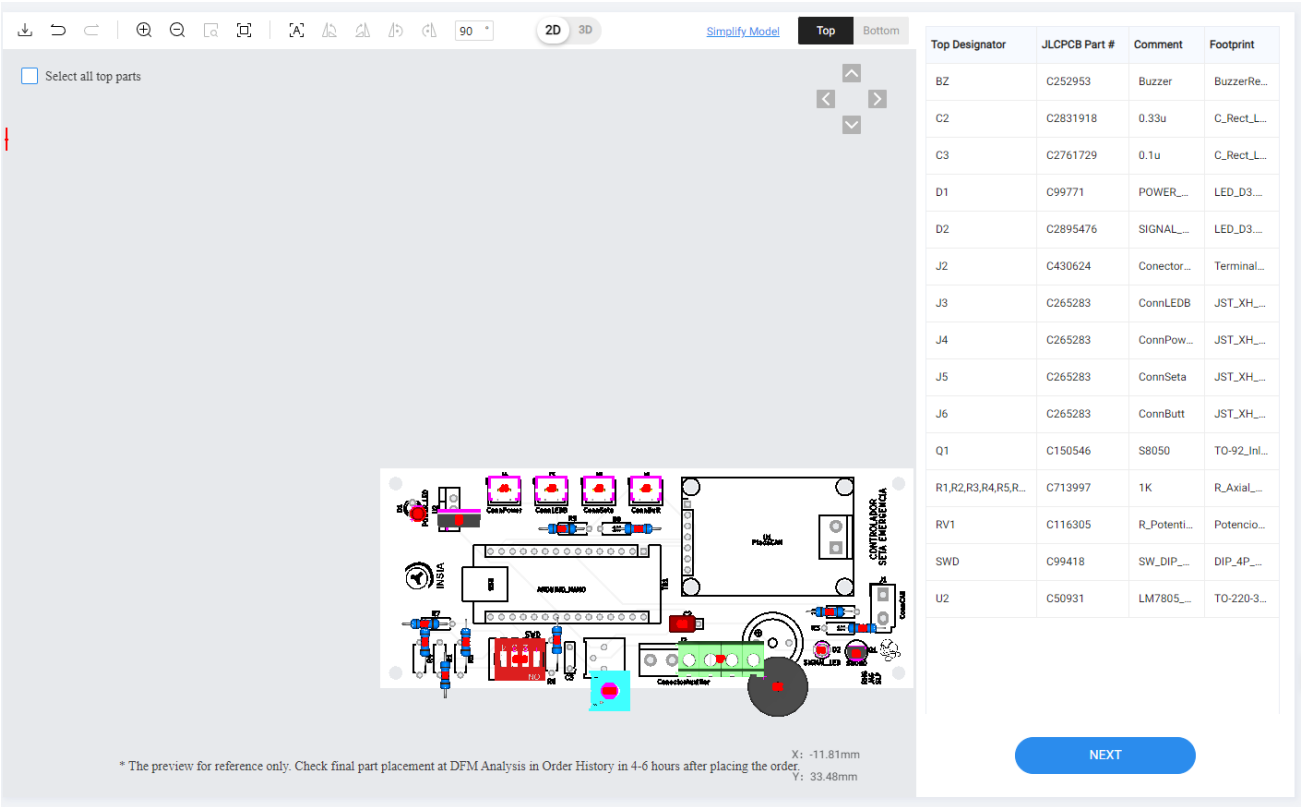


Figura 61. Componentes de la Seta de Emergencia

4.2.4. Montaje de la placa en la página de JCPCB

JCPCB tiene un diseñador para que se puedan colocar los componentes que se han seleccionado en la sección de componentes para que la maquinaria pueda acoplar de manera correcta los componentes en la PCB. Como se puede observar en la siguiente Figura 62, la herramienta de manera automática coloca los componentes de manera incorrecta para que sea el usuario el que se ocupe de revisar todo antes de imprimir.



Top Designator	JLCPCB Part #	Comment	Footprint
BZ	C252953	Buzzer	BuzzerRe...
C2	C2831918	0.33u	C_Rect_L...
C3	C2761729	0.1u	C_Rect_L...
D1	C99771	POWER_...	LED_D3....
D2	C2895476	SIGNAL_...	LED_D3....
J2	C430624	Conector...	Terminal...
J3	C265283	ConnLEDB	JST_XH_...
J4	C265283	ConnPow...	JST_XH_...
J5	C265283	ConnSeta	JST_XH_...
J6	C265283	ConnButt	JST_XH_...
Q1	C150546	S8050	TO-92_Inl...
R1,R2,R3,R4,R5,R...	C713997	1K	R_Axial_...
RV1	C116305	R_Potenti...	Potencio...
SWD	C99418	SW_DIP_...	DIP_4P_...
U2	C50931	LM7805_...	TO-220-3...

Figura 62. Componentes Colocados de forma automática

Por tanto, se procede a colocar todos los componentes en la orientación correcta, importante tener a mano el esquemático de Kicad ya que es de gran ayuda. Finalmente, se tiene el siguiente resultado. En la sección de Anexos se puede observar todas las imágenes con información extra sobre la selección de componentes y el resultado final.

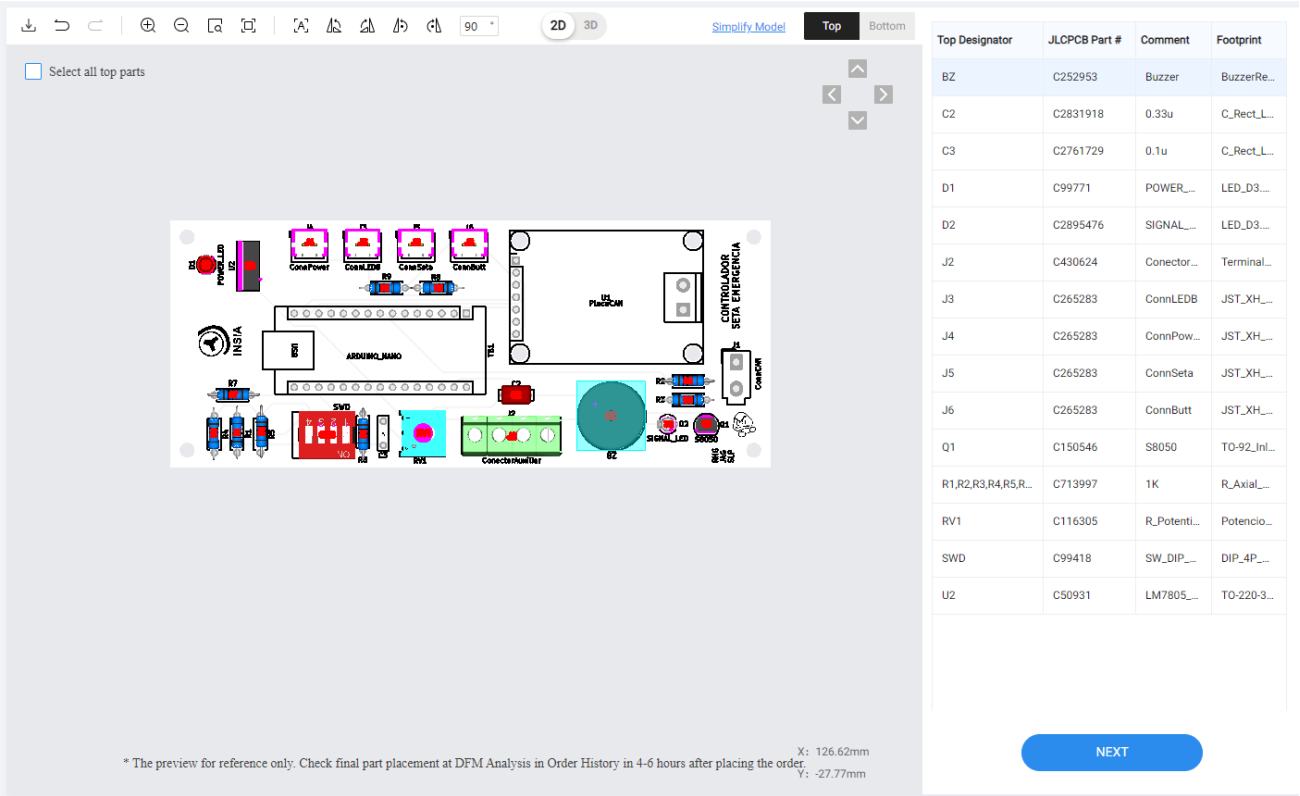


Figura 63. Diseño Final de la Placa en JCPCB

Finalmente se selecciona el color, grosor y las capas que tendrá el material. Se deberá indicar para qué sector va dirigido el proyecto, en este caso Industrial ya que no existe el ámbito académico. Se ha seleccionado la norma ***FR4*** para el proyecto, esta norma está definida por la **NEMA**, es un compuesto de resina epoxídica reforzada de fibra de vidrio. FR tiene como significado en inglés "retardante de llama" indica que el material resiste a la inflamabilidad de los materiales de plástico [26]. Este estándar es la opción más normal para la producción de pequeños lotes de tarjetas o para la creación de prototipos electrónicos.

En la siguiente imagen se muestra la placa nada más recibirla

Figura 64. Placa PCB

4.2.5. Montaje Final y primera prueba

Una vez recibida la placa, se procede a montar los módulos externos que no se incluyeron en la PCB. Estos módulos son el Arduino Nano y la placa MCP2515. Para poder integrar estos módulos externos se ha soldado conectores para los pines, en los cuales se "pinchan" las placas quedando conectadas a las pistas diseñadas en la PCB. Estos conectores son los que se muestran en la Figura 65:

Figura 65. Conectores para Arduino Nano y MCP2515

Y los conectores J1, CANLOW y CANHIGH mostrados en la Figura 66. Estos no se soldaron en JLCPCB ya que se disponían en el INSIA, son fácilmente sustituibles si ocurre algún cortocircuito y son sencillos de reemplazar.

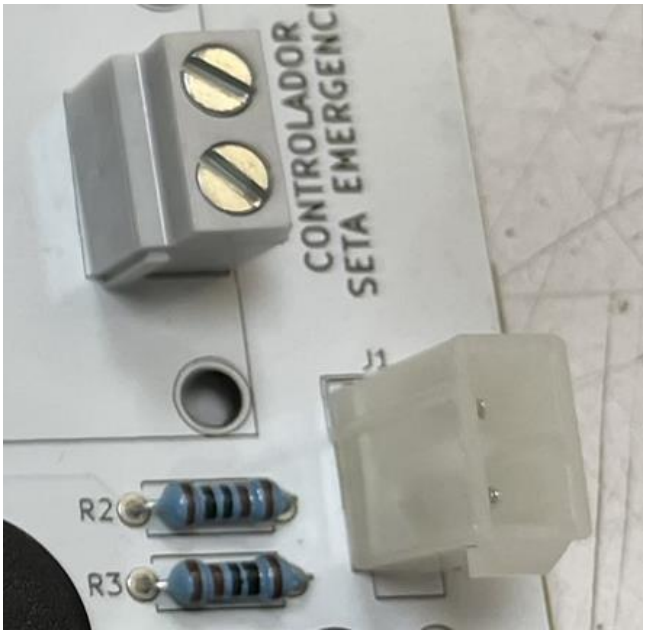


Figura 66. Conectores J1 y CANHIGH, CANLOW

Una vez se tenían todos los conectores instalados, se procedió a instalar el Arduino Nano y la placa MCP2515, comprobar que no había ningún corto en los conectores soldados y realizar cada una de las pruebas unitarias que se explicarán en el 4.5.

Una vez validadas las pruebas unitarias se procedió a cargar el código final a la placa Arduino y comprobar su correcto funcionamiento. Quedando la placa finalizada como se muestra en la siguiente imagen:

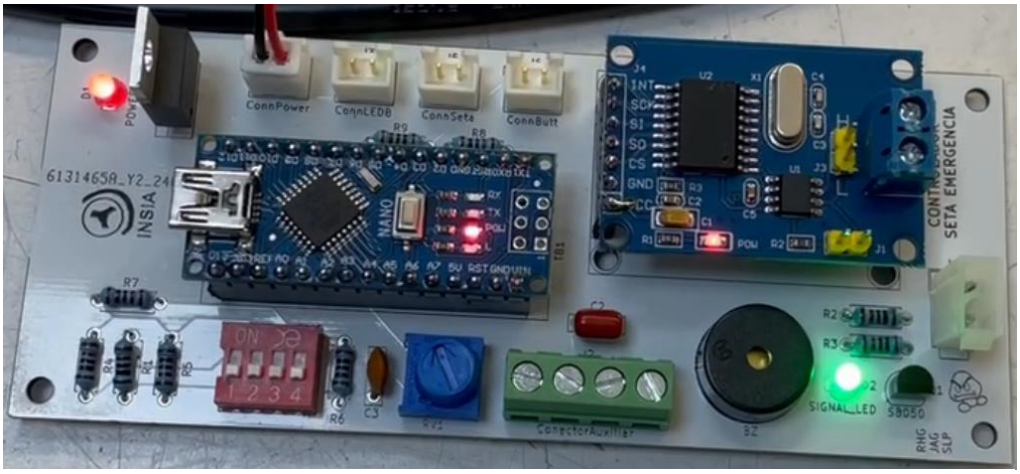


Figura 67. Placa en su estado Final

Para la estructura metálica en la que está montada, se hicieron los agujeros pertinentes para que la placa pueda encajar de manera correcta. Para eso se diseñó una plantilla en Kicad la cual se muestra a continuación.

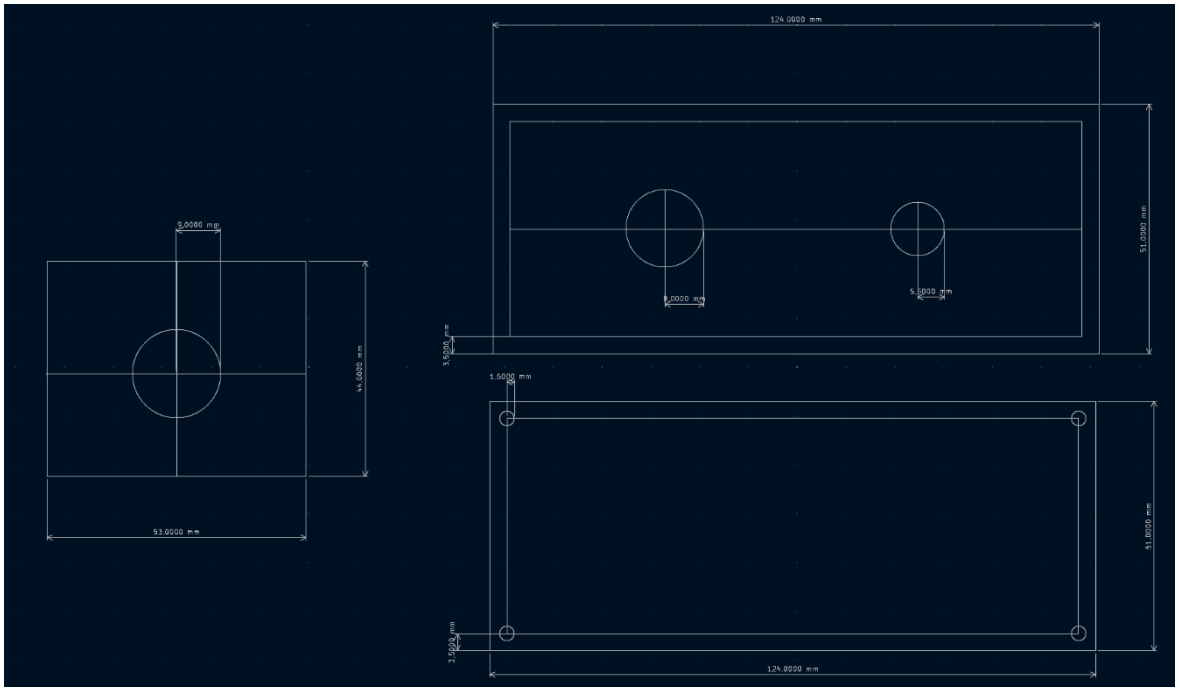


Figura 68. Plantilla para la estructura

Se procedió a imprimir esta plantilla en papel, y se recortó para adherirla momentáneamente en la estructura y proceder a taladrar, asegurándose de que se hacen los agujeros en la posición correcta. Se procedió a montar los switches de la seta de Emergencia y el botonLED. Para que no ocurra ningún corto se introdujo una goma entre la placa y la estructura metálica y se montó la PCB atornillándose con tornillos de métrica 3. Se montó una prensa estopa para poder pasar el cable de la alimentación y el bus CAN al exterior, poder conectarse a otra seta y a la alimentación de 12v. Por último, se atornilló toda la estructura para el acabado final, tomando la forma final mostrada en la Figura 69.

Todo este proceso se repitió por segunda vez para crear una segunda seta de emergencia y demostrar que estas setas pueden funcionar conjuntamente y tienen la posibilidad de enviar su estado en un medio compartido.

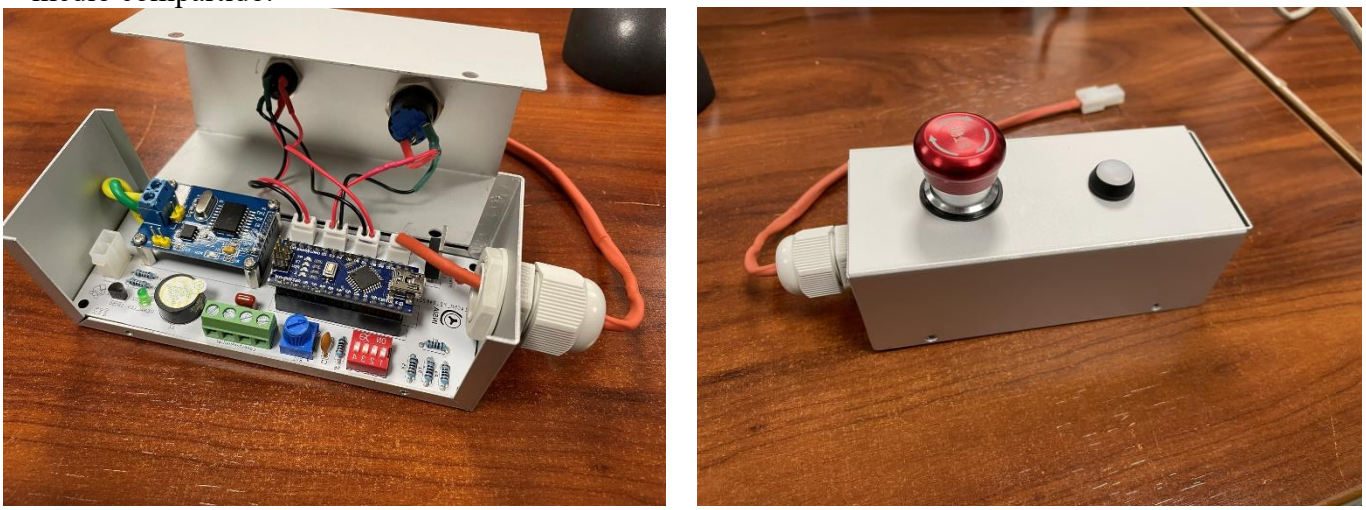


Figura 69. Montaje Final

4.3. SysML

En esta sección se presenta un análisis utilizando SysML, es una extensión de UML que se utiliza para describir sistemas complejos, incluidos aspectos de hardware, software, procesos y comportamiento [27].

SysML permite una presentación visual y conceptual del sistema, lo que facilita la comprensión de su arquitectura y comportamiento. Este enfoque de modelado es útil para identificar y analizar los componentes clave del sistema, así como para comunicar eficazmente su diseño y funcionalidad.

Aunque el uso de este programa no es especialmente sencillo, se ha creado un diseño de alto nivel para exponer el trabajo que se ha realizado, tanto del software como del hardware.

A continuación, se muestra un esquema SysML del sistema de la seta de emergencia en el programa Papyrus:

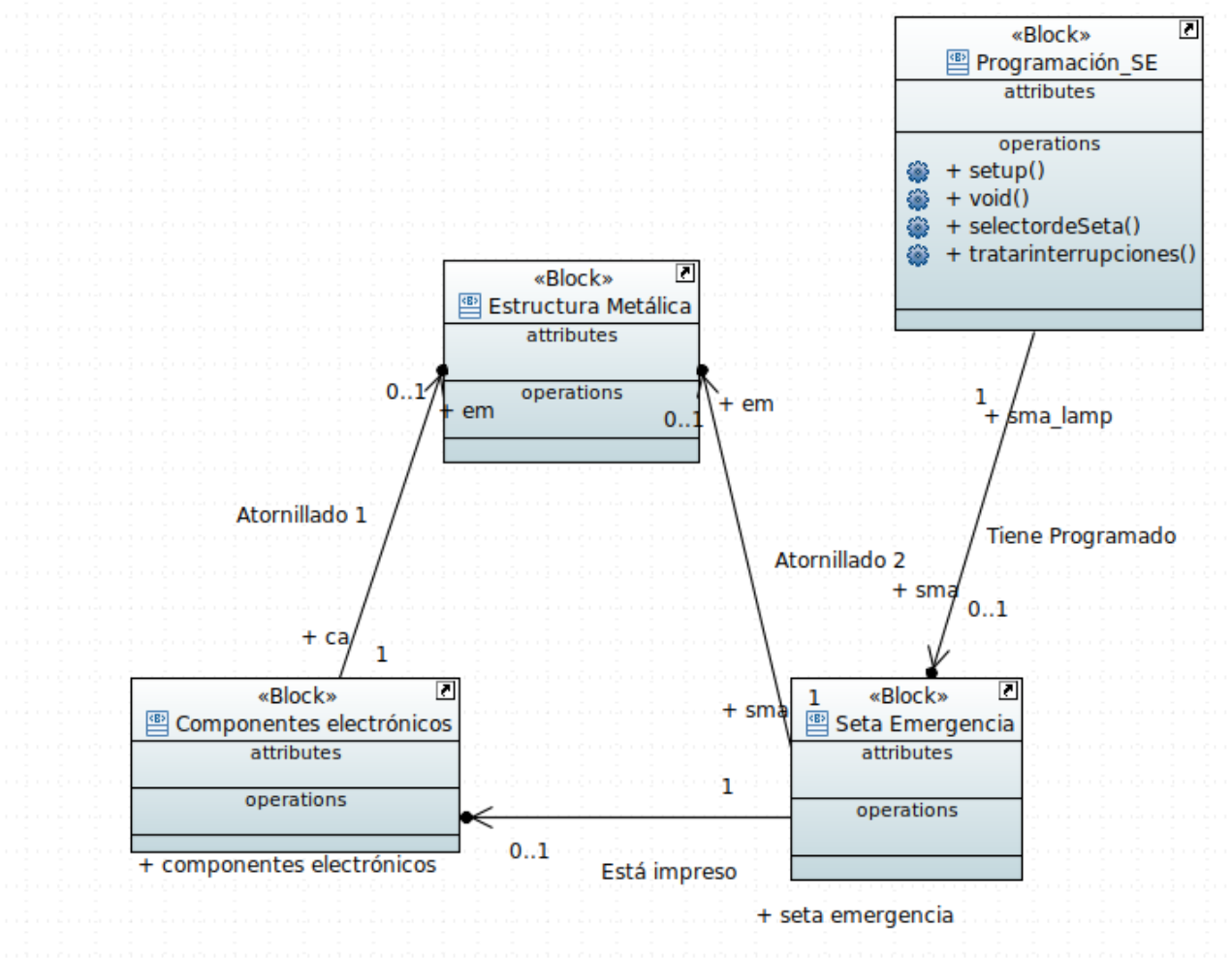


Figura 70. SysML de la Seta de Emergencia

En el esquema anterior, se pueden identificar 4 bloques, representando la estructura metálica en la que se ha montado la seta, la propia instancia de la seta de emergencia, el código programado y la PCB diseñada. Además, se muestran las relaciones y conexiones entre estos elementos, lo que proporciona una visión general de la estructura y funcionamiento del sistema.

4.4. Máquina de estados

Para una mayor aclaración de lo que el sistema es capaz de realizar, se ha creado una máquina de estados como se puede ver en la siguiente Figura 71:

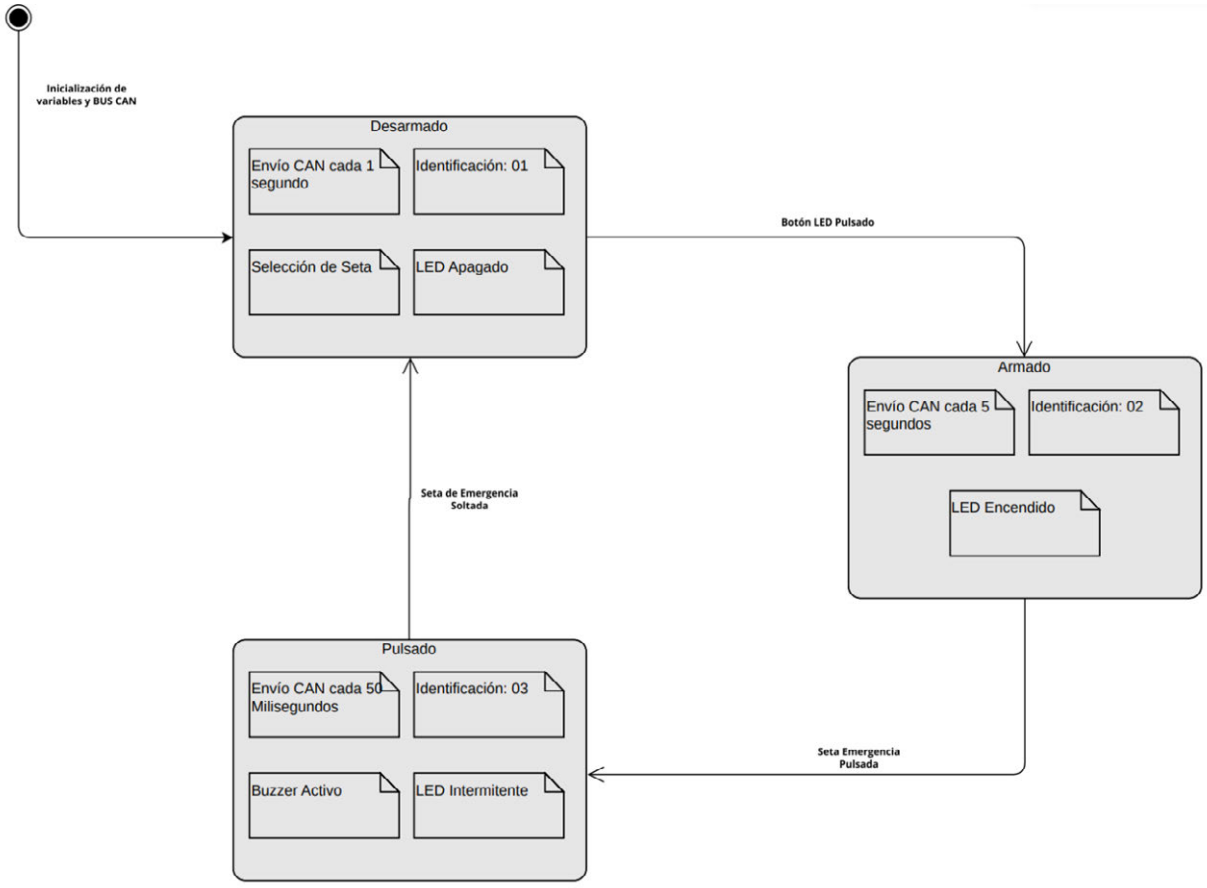


Figura 71. Máquina de Estados

En un primer momento el sistema se inicializa en el estado *Desarmado*, por tanto, el LED estará apagado, se recogerá qué seta de Emergencia está Pulsada y finalmente se enviará el mensaje CAN programado cada *1 segundo* con identificador *01*.
Para poder cambiar de Estado se debe de pulsar el BotonLED integrado en la SE, una vez pulsado se transita al estado *Armado* que enciende el LED del Switch y envía un mensaje CAN cada *5 segundos* con identificador *02*. Para poder transitar al estado *Pulsado* se debe de pulsar el switch de la Seta de Emergencia. Aquí se apagará el LED del BotonLED y se enviará un mensaje CAN cada *50 ms* con identificador *03*.

Una vez en este estado solo es posible transitar al estado *Desarmado* volviendo al estado que se ha explicado al principio, siendo posible seleccionar la seta que se desee modificar.
Como se puede observar, esta máquina de Estados no tiene un punto final, y eso es así ya que este sistema está en continuo funcionamiento y no se contempla que tenga una desconexión premeditada, sino que la desconexión sería de manera repentina.

4.5. Pruebas unitarias del código programado

Para comprobar el correcto funcionamiento del Hardware y Software utilizado, se han validado cinco pruebas unitarias. Se podrá comprobar el código en la sección **Anexos**, por tanto, se procede a explicar cada una de ellas.

- Prueba unitaria Switch BotonLED, Switch Seta Emergencia
  Se ha programado una función que simplemente muestra por pantalla si el switch correspondiente ha sido pulsado o no.

- Prueba unitaria DipSwitch
  Esta prueba es similar a la de los Switches, la diferencia es que esta es la encargada de seleccionar la seta y se debe de verificar si su funcionamiento es correcto. Por tanto, se seleccionan los pines digitales correspondientes a la selección de setas y se muestra por pantalla el resultado.

- Prueba unitaria LED
  Se procede a encender el led en un período de tiempo establecido, en este caso 1 segundo, por tanto, si el Hardware es correcto y todo está conectado sin ningún cortocircuito el LED se encenderá durante 1 segundo y permanecerá apagado durante otro segundo.

- Prueba unitaria CAN MCP2515
  Esta prueba consiste en comprobar si la placa MCP2515 consigue enviar de forma correcta la trama de datos CAN con el formato CANOpen utilizado en el código final. Se inicializa el serial y el BitRate como se ha explicado en la sección 4.1, y se comprueba con el Peak y utilizando el programa PCANView si el mensaje ha sido enviado correctamente, de la misma forma con la petición de mensaje CAN.

- Prueba unitaria buzzer
  Se ha probado el buzzer que incorpora la seta, esto se ha llevado a cabo con una librería de Arduino llamada Tone [28], la cual gracias a sus funciones es capaz de generar frecuencias para que el buzzer pueda sonar en distintos tonos. Aunque el buzzer que se incorpora es activo, se puede hacer sonar con el envío de una señal no tonificada. Por tanto se ha procedido a hacer la prueba de la señal con tono y la señal con tono en el buzzer. Para el primero simplemente se utiliza la función *freq.play* siendo freq un vector de valores entre el 31 al 4798. Para la segunda prueba simplemente se envía una señal High al pin correspondiente del buzzer, en este caso el pin digital 5, y se trata la señal con la función millis para que se encienda y se apague cada un segundo.

5. Resultados Obtenidos

Para analizar los resultados obtenidos a lo largo del proyecto se han realizado pruebas para comprobar su funcionamiento correcto así como su durabilidad en el tiempo. Esto sirve sobre todo para comprobar si los componentes utilizados y la PCB diseñada funcionan de manera correcta. A continuación se describiran cada una de las pruebas y cómo han funcionado.

5.1. Metodología

Una vez validadas las pruebas unitarias del Hardware y de las funciones principales utilizadas, se procede a comprobar el correcto funcionamiento del sistema en conjunto. Para eso se ha ido programando a lo largo del tiempo y comprobando su funcionamiento, si no era así, se corregía el código pertinente hasta que la función fuese la que realmente se quería. Así fue tanto con el envío de mensajes CAN y con el DipSwitch cambiando el número de seta únicamente en el estado Desarmado.

Por tanto, se ha seguido el método en cascada para el desarrollo del Software en el sistema, este método también conocido como modelo de desarrollo en secuencia. Este enfoque de desarrollo de software se caracteriza por una secuencia lineal y jerárquica de etapas, en las cuales cada fase debe de completarse antes de avanzar a la siguiente. A continuación, se detalla cómo se aplicó este método en el proyecto y se relaciona con la experiencia adquirida durante su desarrollo:

1. Especificación de requisitos: En esta fase inicial, se identificación y definieron los requisitos del sistema, incluyendo las funcionalidades y características que debía de cumplir la seta.

2. Diseño del sistema: Se elaboró el diseño detallado del sistema, considerando la arquitectura Hardware y Software para cumplir los requisitos establecidos. Se decidió utilizar el arduino Nano y la placa MCP2515 para el Hardware principal y el software platformIO como se ha explicado en el apartado 3.1.

3. Implentación: Se desarrolló el código del sistema así como el diseño del Hardware para la PCB.

4. Integración y Mantenimiento: Por último, se integraron todas las partes del sistema y se realizaron las pruebas unitarias para garantizar su correcto funcionamiento en conjunto.

Durante el desarrollo del proyecto, la aplicación del método en cascada permitió una gestión ordenada y estructurada de las actividades, facilitando la planificación y el seguimiento del proceso. Se ha obtenido una valiosa experiencia en cada etapa, destacando los siguientes aspectos:

- Claridad en los Requisitos: La fase de especificación de requisitos permitió una comprensión clara de las necesidades y expectativas del sistema, proporcionando una base para el diseño y la implementación

- Organización: El enfoque secuencial del método en cascada facilitó la implementación gradual del sistema, permitiendo resolver los problemas surgidos y realizar ajustes a medida que avanzaba el proyecto.

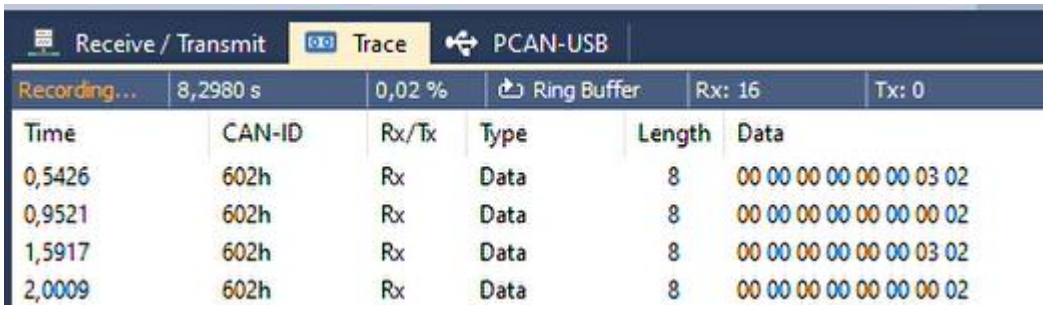
- Verificación: Se llevaron a cabo las pruebas unitarias verificando que todo funcionase de froma

correcto tanto del Software como del Hardware.

- Adaptaciñon y Continuidad: Se contempló la posibilidad de realizar mejoras y ajustes en el sistema, en función de los resultados de las pruebas y la retroalimentación recibida, asegurando la calidad y eficacia del producto final.

5.2. Mensajes CAN

A continuación se expondrán los diferentes mensajes CAN que se han obtenido realizando diferentes pruebas en el sistema. Se disponen de dos setas de emergencia, una con el número de seta 0 y la otra con el número 3

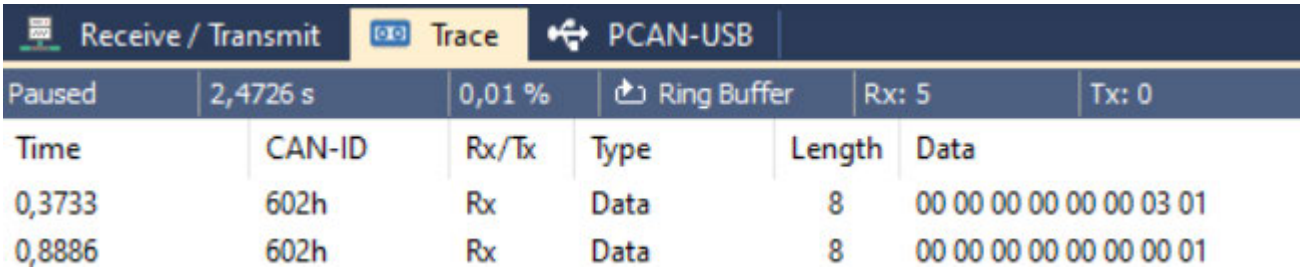

Receive / Transmit | Trace | PCAN-USB

Recording... | 8,2980 s | 0,02 % | Ring Buffer | Rx: 16 | Tx: 0

Time	CAN-ID	Rx/Tx	Type	Length	Data
0,5426	602h	Rx	Data	8	00 00 00 00 00 00 03 02
0,9521	602h	Rx	Data	8	00 00 00 00 00 00 00 02
1,5917	602h	Rx	Data	8	00 00 00 00 00 00 03 02
2,0009	602h	Rx	Data	8	00 00 00 00 00 00 00 02

Figura 72. Mensaje CAN 1

En un primer instante, tanto la seta 0 como la 3 se inicializan en el estado Desarmado, mostrando el 01 en el byte 7 del campo de datos. Enviando su mensaje CAN cada segundo.

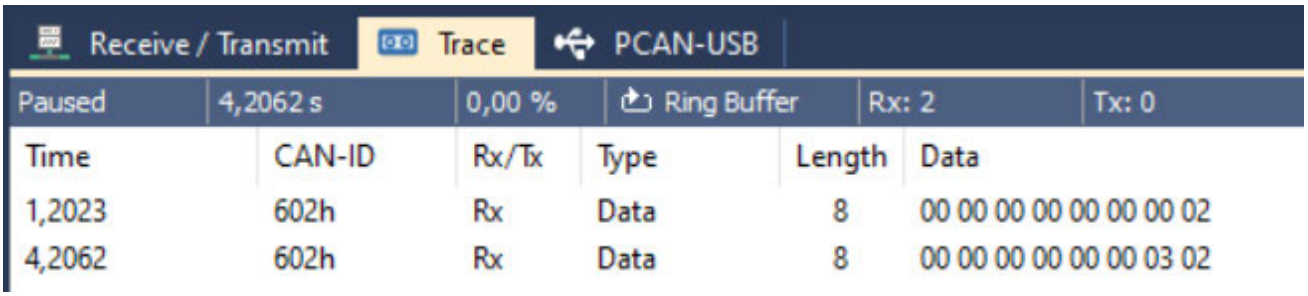

Receive / Transmit | Trace | PCAN-USB

Paused | 2,4726 s | 0,01 % | Ring Buffer | Rx: 5 | Tx: 0

Time	CAN-ID	Rx/Tx	Type	Length	Data
0,3733	602h	Rx	Data	8	00 00 00 00 00 00 03 01
0,8886	602h	Rx	Data	8	00 00 00 00 00 00 00 01

Figura 73. Mensaje CAN 2

Seguidamente se pulsa el botón LED para armar el sistema, por tanto el mensaje que se enviará al bus será el mismo que el anterior pero cambiando el último byte a 02.


Receive / Transmit | Trace | PCAN-USB

Paused | 4,2062 s | 0,00 % | Ring Buffer | Rx: 2 | Tx: 0

Time	CAN-ID	Rx/Tx	Type	Length	Data
1,2023	602h	Rx	Data	8	00 00 00 00 00 00 00 02
4,2062	602h	Rx	Data	8	00 00 00 00 00 00 03 02

Figura 74. Mensaje CAN 3

El mensaje contendrá 03 en el byte 7 cuando el estado sea pulsado.

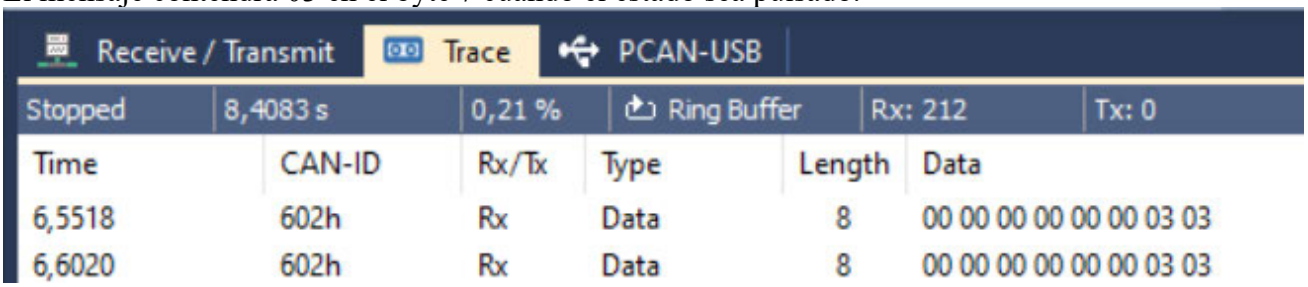


Figura 75. Mensaje CAN 4

Como se explicó en el apartado 4.1.2, el sistema es capaz de enviar un mensaje de respuesta cuando el usuario hace una petición del estado del sistema, el cual debe de tener un 20 en el byte 1 del campo de datos del mensaje CAN. En la siguiente figura se muestra cómo se recibe el mensaje de respuesta conteniendo en el byte 0 del campo de datos un 40 y cuál es el mensaje de petición que se envía.

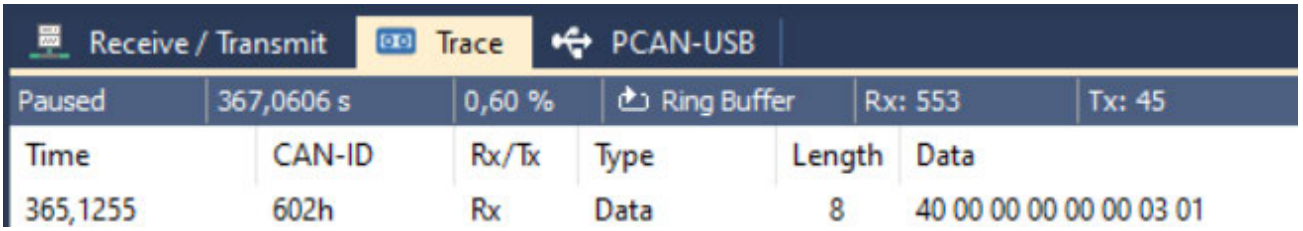


Figura 76. Mensaje CAN 5

5.3. Consumo de Energía

En este apartado se estudiará el consumo de Energía del sistema, con el uso de una tabla de energía en la que se insertan los consumos medios y máximos de cada uno de los componentes. De esta forma se es capaz de calcular los Amperios que consumirá con la siguiente fórmula:

$$\sum(PeakConsumption5.5V, PeakComsumption3.3V)$$

La energía Consumida será de 119mA en 5,5V utilizando el inversor de potencia LM7805 mencionado en el apartado 4.2.3

Consumo de Corriente del Sistema

Total System Current Draw

Device	Datasheet	Voltage (V)	Average Consumption (mA)	Peak Consumption (mA)	Rate of system usage
Arduino Nano	https://docs	5	15	19	16,67%
Power LED	https://www	5	20	20	17,54%
Power LED	https://www	5	20	20	17,54%
Power LED	https://www	5	20	20	17,54%
Buzzer	https://file.e	12	30	30	26,32%
MCP2515	https://ww1	3,3	5	5	4,39%
	mA	**12V**	**5,5V**	**3,3V**	
Average total draw.Consumo Total	110	30	110	0	
Peak total draw. Consumo Total Máximo	114	30	114	5	

Energía Consumida (mA) 119

5.4. Registro de Eventos con Diagrama de Tiempo

Para conocer mejor el funcionamiento del sistema, se proporcionará a continuación una tabla que representa un escenario real del sistema, en donde se puede ver los cambios de estado que el usuario ha realizado con sus respectivos mensajes CAN

Switch Pulsado	ESTADO	MENSAJES CAN en MiliSegundos
-	Inicialización	
Boton LED	Desarmado	**1000**
	Armado	**5000**
Seta Emergencia	Pulsado	**50**
-	Pulsado	**50**
-	Pulsado	**50**
-	Pulsado	**50**
-	Pulsado	**50**
-	Pulsado	**50**
Boton LED	Pulsado	**50**
-	Pulsado	**50**
-	Pulsado	**50**
Soltado Seta Emergencia	Desarmado	**1000**
-	Desarmado	**1000**
Seta Emergencia	Desarmado	**1000**
-	Desarmado	**1000**
Boton LED	Armado	**5000**
-	Armado	**5000**
Boton LED	Armado	**5000**
Seta Emergencia	Pulsado	**50**
-	Pulsado	**50**
-	Pulsado	**50**

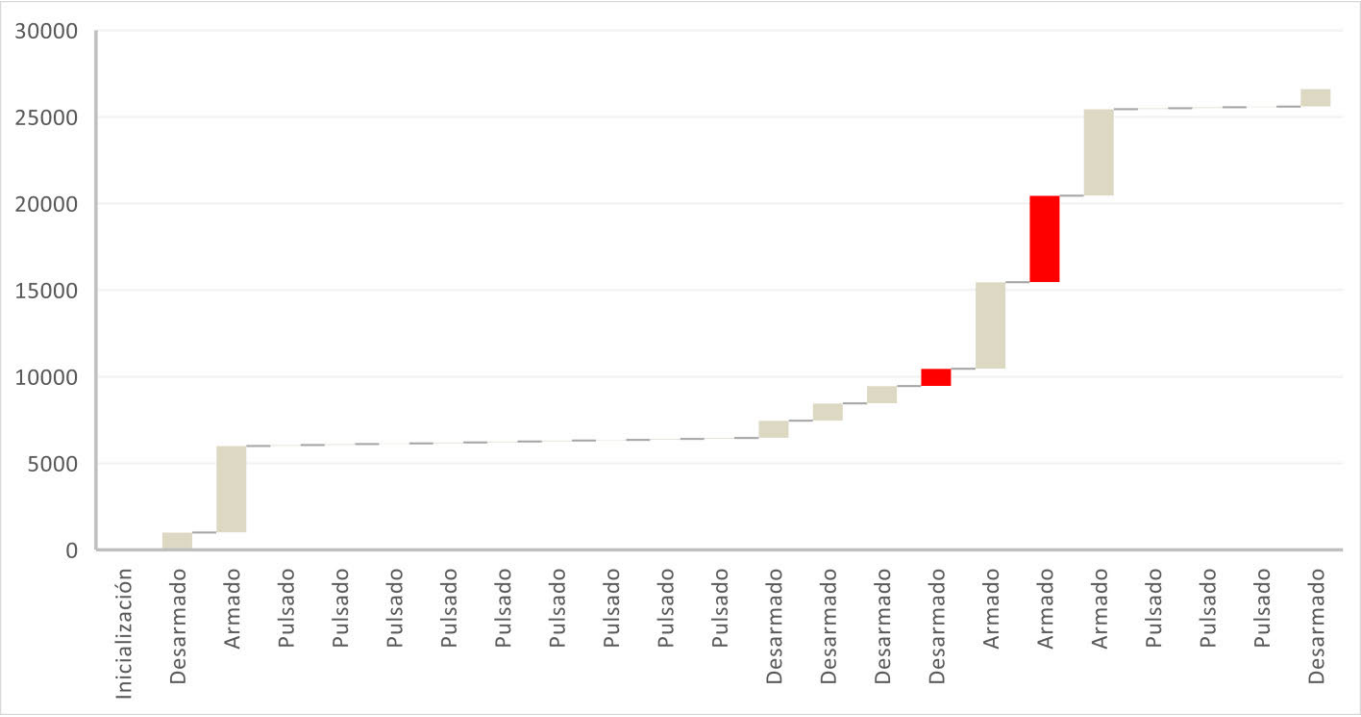


Tabla 2. Diagrama de tiempo y Registro de Evento

Como se puede observar, en un primer instante se inicaliza el sistema en el estado *Desarmado*, se pulsa el Boton LED para Armar el sistema e ir al estado *Armado*, seguidamente se pulsa el Switch de la Seta de Emergencia transitando al estado *Pulsado*, y se pulsa de nuevo el Boton LED pero como no se contempla esta acción y no se transita a ningún estado ya que primeramente se tiene que despulsar el Switch de la Seta de Emergencia para transitar al estado *Desarmado* como se hace después, es decir los estados en color rojo son estados fallidos que el usuario ha pulsado erróneamente sin seguir el transcurso correcto del sistema. Seguidamente se pulsa el switch de la seta de emergencia pero como ese no es recorrido el sistema sigue en el estado *Desarmado* como si no se hubiese pulsado ningún switch. Seguidamente se pulsa el botón LED y por tanto se transita al estado *Armado*. Finalmente se pulsa la seta de emergencia volviendo al estado *Pulsado* y soltando la seta de emergencia volviendo al estado *Desarmado*.

6. Conclusiones y Trabajos Futuros

En esta última sección del proyecto se centrará en las conclusiones que se han logrado obtener y los futuros trabajos en los que se pueden continuar para llevar este sistema a un más alto nivel, exponiendo los problemas encontrados y las soluciones que se han llevado a cabo para su correcta continuación.

6.1. Conclusiones

En este proyecto se ha mostrado cómo se ha desarrollado la creación de una seta de emergencia desde el principio hasta el producto final, pasando por una lluvia de ideas para identificar los requisitos, teniendo una mayor precisión en cuanto a las características del sistema, la programación de todas las funcionalidades y las pruebas que se han llevado a cabo para validarlas, el diseño de la placa PCB y el proceso para la impresión de la misma con la compañía JLCPCB, el montaje de todos los componentes externos de la placa con sus respectivas soldaduras y la prueba del sistema en su estado final.

Este trabajo ha sido de gran ayuda en cuanto a la compresión de la elaboración de un producto desde el principio hasta el final, en relación con la asignatura de Sistemas Empotrados. Obteniendo diferentes conocimientos tanto de programación, diseño de placas, validación y software, así como la práctica del método en cascada utilizado en el desarrollo del proyecto.

De este modo el sistema es capaz de enviar y recibir mensajes CAN, cambiar de estados según los switches pulsados mostrando un aviso tanto visual a través de LEDs, como sonoro gracias al buzzer que se incorpora, selección del número de setas con el objetivo de generar un sistema con capacidad de interconexión en el vehículo. Todo esto diseñado exclusivamente para este proyecto e impreso en PCB gracias a la compañía JLCPCB, integrado y montado en una estructura robusta teniendo como producto final un sistema de seguridad con un acabado compacto y funcional.

Con este proyecto se ha podido avanzar en diferentes sectores tanto del mundo académico como del profesional. Obtener conocimientos no solo de programación y diseño, sino de destrezas importantes en el sector de la ingeniería como el uso de taladros, soldadores, sierras y lijadoras, importantes en la creación física del proyecto, no solo estudiando y progruesando en el ámbito digital. Además, se ha obtenido personalmente una gran experiencia en cuanto a los sistemas empotrados y a los sistemas en tiempo real resolviendo los problemas que han surgido de estos sectores y que han sido satisfactoriamente resueltos, con la ayuda de los compañeros y profesores que se ha tenido.

En conclusión, este proyecto no solo ha sido de utilidad para la elaboración de un trabajo de fin de grado, sino que ha permitido la integración de diferentes ideas y proyectos en el INSIA, la aportación de conocimientos y experiencia en el sector y la pequeña aportación en el sector automovilístico en cuanto a la seguridad que este sistema puede proporcionar tanto en la vida cotidina como en la profesional.

6.2. Trabajos Futuros

En cuanto a la continuación del sistema, se han llevado a cabo procesos que han tenido una dificultad elevada, como el diseño de la PCB y el proceso de montaje del mismo, en los que se pueden añadir diferentes funcionalidades que pueden crear un gran proseguimiento al sistema. A continuación se detallarán diferentes continualidades que se han presentado tanto en el transcurso de la elaboración del sistema como en su final y que no se han podido integrar debido a la falta de tiempo y la alta complejidad que pueden presentar.

- **Elaboración de más setas de emergencia**

Aunque se han elaborado dos setas de emergencia, pudiendo ver cómo es posible la interconexión entre ellas, la elaboración de más sistemas puede permitir la visualización de cómo el protocolo CAN puede enviar varios mensajes en un mismo bus sin que haya ninguna colisión de los mensajes y por tanto que esta transmisión de información sea coherente.

- **Integración de relés para una mayor seguridad**

La incorporación de relés al sistema podría proporcionar una capa de seguridad extra al eliminar las depencias que tiene el sistema con un computador dentro del vehículo. Esto permitiría que la seta de emergencia lleve a cabo de forma independiente determinadas acciones, como cortar la alimenación o activar los frenos, sin necesidad de intervención externa.

- **Estructura metálica resistente al polvo y agua**

En su aplicación en entornos de exteriores o en vehículos que no tengan un habitáculo cerrado, se propone la implementación de una estructura resistente al polvo y al agua, esto garantiza la durabilidad y fiabilidad del sistema incluso en condiciones climáticas desfavorables o entornos adversos.

- **Mejoras en el código**

Para mejorar el rendimiento del sistema, se podría mejorar líneas de código que permitirían aumentar su eficiencia y garantizar su fiabilidad a largo plazo. Esto puede incluir, tratos en el cambio de las excepciones y un mejor tratamiento de las interrupciones pudiendo transitar a un estado de error si es el caso. Esta mejora del código puede mejorar la experiencia del usuario.

- **Integración de sistemas de refrigeración más sofisticados**

Aunque se ha estudiado la disipación de calor en una temperatura en torno a los 26 grados, y se ha decidido que con el consumo del sistema no es necesario una refrigeración extra si que cabe la posibilidad de la mejora de su refrigeración pudiendo incorporar sistemas de refrigeración más potentes lo que puede garantizar el correcto funcionamiento en condiciones de temperaturas extremas. Esto puede incluir la integración de sistemas de refrigeración activos o pasivos, como la incorporación de un disipador más grande o un ventilador, para mantener los componentes electrónicos dentro de rangos de temperatura seguros y óptimos.

6.3. Problemas encontrados

Durante el proceso de desarrollo del proyecto, se ha enfrentado a diversos problemas y obstáculos que han requerido soluciones para lograr la funcionalidad correcta del sistema. A continuación, se detallan los problemas encontrados y las acciones tomadas para solucionarlos:

- **Cambio del Buzzer**

Se identificó un problema con el buzzer incorporado en la PCB, el cual no sonaba tan alto como se esperaba. El problema fue debido al buzzer que se incorporó en JLCPCB el cual en el datasheet del buzzer mostraba que tení un rango de hasta 32v y que llegaba a su máximo en 12v. Sin embargo al hacer las pruebas con este buzzer y estableciendo el valor del potenciómetro al mínimo, el buzzer sonaba muy bajo. El sistema funciona a 12v que le llegan directamente al buzzer, sin embargo se llegó a la conclusión de que el buzzer incorporado en el sistema, da su máximo sonido a más voltaje. Por tanto se procedió a hacer una prueba con otro buzzer que se disponía en el laboratio, pinchando en las terminales en las que se conectaba el antiguo buzzer. Como era de esperar el buzzer nuevo sonaba en el nivel que se tenía pensado y por tanto se procedió a desoldar el buzzer antiguo y soldar el nuevo en la placa.

- **Cambio de Switch de la Seta de Emergencia**

Este ha sido un problema menor, ya que al tomar las medidas, no se tubo en cuenta cuánto medía el switch de la seta de emergencia y este era demasiado grande. Por tanto al montar el sistema en la estructura metálica, no cerraba del todo ya que el hueco de alto que tenía la estructura era insuficiente y tocaba los pines del switch con el Arduino, por consiguiente no se podía cerrar la estructura obteniendo un resultado no profesional.
La solución fue integrar un switch de seta de emergencia más pequeño, de esta forma se tenía mayor hueco y ya se podía encajar todo de forma correcta.

- **Problema con pines de Interrupción**

Para la espera de los mensajes CAN programados, se han programado interrupciones, sin embargo, no todos los pines del Arduino Nano permiten una interrupción externa, los pines que sí lo permiten son los digitales 1 y 2. Por lo que se cambiaron los pines que anteriormente estaban en el digital 9 y 10, y reposicionando correctamente los demás pines que se utilizan para las demás funciones del sistema, solucionando de esta forma el problema.

- **Problema con mensajes CAN**

Cuando ya se tenía avanzado el proyecto y se estaba enviando los mensajes CAN correspondientes, en ciertas ocasiones los mensajes CAN no se lograban transmitir y por consiguiente no se veían reflejados en el software de PCANView. Se llegó a la conclusión de que el sistema consumía demasiado para los 5v que se proporcionaba desde el Arduino conectado al USB del computador. Sin embargo, haciendo diferentes pruebas con diferentes escenarios con menor consumo, quitando algunas resistencias y switches, el problema persistía.

Por tanto se procedió a cambiar la manera en la que se transmitían los mensajes CAN utilizando interrupciones con el switch y utiliazando un índice de datos más sofisticado, cambiando cuándo se tenían que enviar estos según se pulsaban los
Switches. De esta manera se pudo solucionar este problema, pudiendo enviar los mensajes CAN cuando correspondía y sin dar ningún error en la transmión y recepción de los mensajes

6.4. Planificación Temporal y presupuesto

6.4.1. Planificación

A continuación se muestra el diagrama de tiempo utilizando el método en cascada.

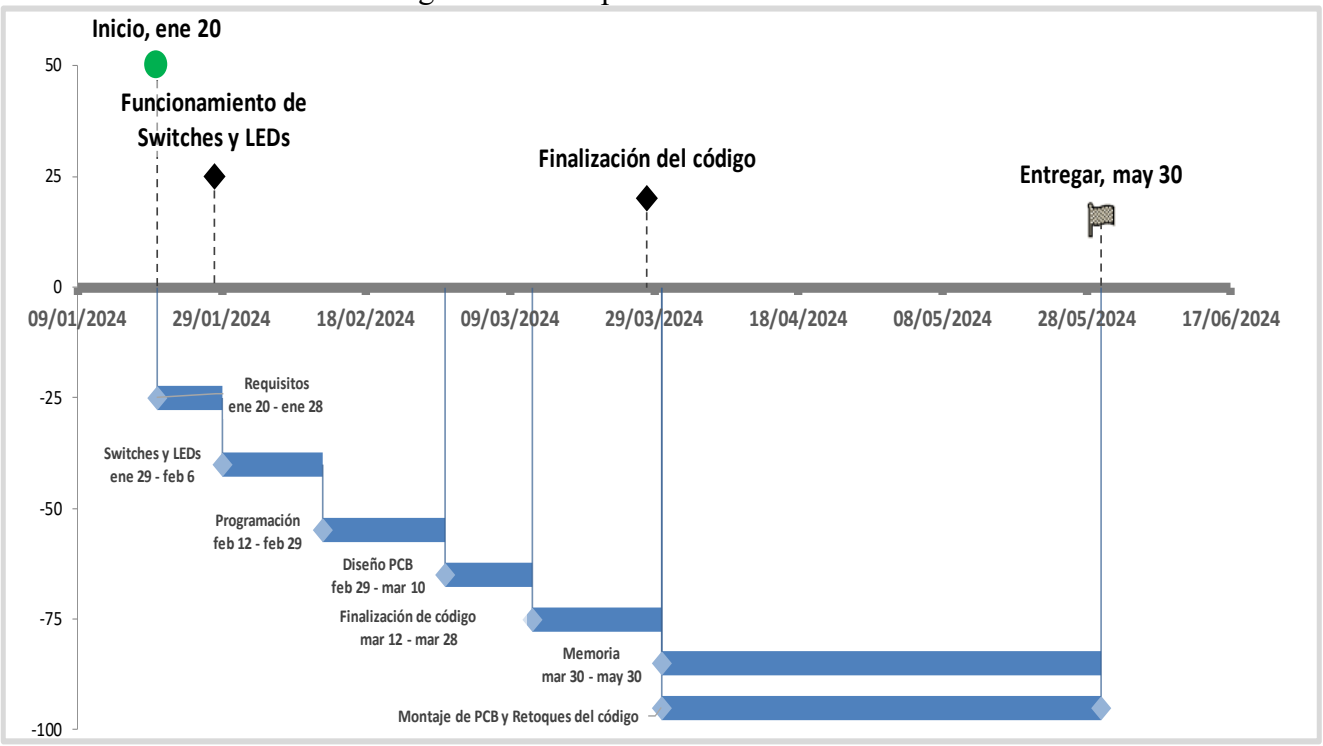


Tabla 3. Cronograma

6.4.2. Presupuesto

Se mostrarán en la siguiente tabla los gastos totales del proyecto realizado. Se ha utilizado software de código libre, es decir que su descarga y uso es totalmente gratuita, estos programas han sido los ya mencionados en el apartado 3.1, Kicad, platformIO, VisualStudio, PCANView, Papyrus y GitHub.

TFG

PRESUPUESTO

GASTO	IMPORTE
Switches	4,00 €
Estructura Metálica	20,00 €
BotonLED x2	40,00 €
PCB x5	60,03 €
MCP2515 x2	11,69 €
Arduino Nano x2	16,99 €
Soldador y Estaño	14,44 €
Horas de desarrollador	2.500,00 €
Total	**2.667,15 €**

Tabla 4. Presupuesto

6.5. Impacto Social y Ambiental

Estos sistemas de seguridad afectan de mayor o menor forma a la sociedad y el medio ambiente. La implementación de la seta de emergencia contribuye a mejorar la seguridad vial al proporcionar una señal de advertencia clara y visible en situaciones de emergencia en la carretera. Esto puede ayudar a prevenir accidentes y proteger la vida de los conductores, pasajeros y peatones, alertando a otros conductores sobre situaciones de emergencia, como averías o accidentes en la carretera. La seta de emergencia puede ayudar a reducir la probabilidad de colisiones cuando se está utilizando un vehículo de pruebas en las que puede suceder algún problema en el vehículo.

La incorporación de tecnologías de seguridad activa en los vehículos como es la seta de emergencia puede servir como herramienta educativa para concienciar a los conductores sobre la importancia de la seguridad vial y fomentar conductas responsables en la carretera.

En cuanto al impacto Ambiental, al mejorar la seguridad vial y reducir la incidencia, este proyecto contribuye a la reducción de emisiones contaminantes y la congestión del tráfico. Menos accidentes significan menor tiempo de inactividad en la carretera, lo que se traduce en una menor cantidad de emisiones de gases de escape.

La incorporación de tecnologías como esta, ayuda a promover un sistema de transporte más seguro y eficiente que fomenta el uso de medios de transporte más respetuosos con el medio ambiente. El uso eficiente de recursos en el diseño y la implementación , tanto en términos de hardware como software, contribuyen a un uso más racional de los recursos naturales y energéticos, minimizando el impacto ambiental asociado con su producción, operación y disposición final.

Es decir, este proyecto no solo tiene beneficios técnicos, sino que también tiene un impacto positivo en la sociedad al mejorar la seguridad vial, en el medio ambiente y no solo en el sector automovilístico sino en otros sectores de seguridad para la población, como por ejemplo, en los ascensores en edificios, en el control de aviones y trenes, y al parar en situaciones de emergencia en cualquier cinta transportadora. Estos aspectos sociales y ambientales son fundamentales para el desarrollo y la implementación de tecnologías innovadora que buscan mejorar la calidad de las personas y proteger el entorno natural.

7. Bibliografía

[1] Mirada Profunda A La Historia De Seguridad De Los Automóviles. Accedido: 26 de marzo de 2024. Disponible en: https://arashlaw.com/es/una-mirada-profunda-a-la-historia-de-seguridad-de-los-automoviles/

[2] "Father of driver's education" was a Penn State professor. Accedido: 26 de marzo de 2024. Disponible en: https://onwardstate.com/2014/09/26/history-lesson-the-national-birth-of-driver-education-at-penn-state/

[3] Diferencia entre un Cristal Templado y Laminado. Accedido: 26 de marzo de 2024. Disponible en: https://www.carglass.es/blog/conduce-seguro/cual-es-la-diferencia-entre-cristal-laminado-y-cristal-templado/

[4] Quién inventó el airbag. Accedido: 26 de marzo de 2024. Disponible en: https://okdiario.com/ciencia/quien-invento-airbag-8858914

[5] S. Embebidos, «Introducción al bus CAN».

[6] Bus CAN - Wikipedia, la enciclopedia libre. Accedido: 27 de marzo de 2024. Disponible en: https://es.wikipedia.org/wiki/Bus_CAN

[7] Protocolo de capa de aplicación CAN: CANopen. Accedido: 27 de marzo de 2024. Disponible en: https://www.logicbus.com.mx/blog/canopen/

[8] SISTEMA MOST-BUS. Accedido: 28 de marzo de 2024. Disponible en: https://issuu.com/aritza2006/docs/can-bus_1_/s/23312571

[9] FlexRay Automotive Communication Bus Overview. Accedido: 28 de marzo de 2024. Disponible en: https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/flexray-automotive-communication-bus-overview.html

[10] FlexRay. Accedido: 28 de marzo de 2024. Disponible en: https://es.wikipedia.org/wiki/FlexRay

[11] Cómo funciona el Sistema de diagnóstico OBD. Accedido: 13 de abril de 2024. Disponible en: https://www.motor.mapfre.es/consejos-practicos/consejos-de-mantenimiento/como-funciona-el-sistema-obd/

[12] EOBD. Accedido: 28 de marzo de 2024. Disponible en: https://www.diariomotor.com/que-es/eobd/

[13] OBD2. Accedido: 28 de marzo de 2024. Disponible en: https://www.race.es/que-es-obd-coche

[14] Conectados para la seguridad: El papel transformador de las comunicaciones V2V y V2I. Accedido: 28 de marzo de 2024. Disponible en: https://www.tomorrow.bio/es/post/connected-for-safety-el-papel-transformador-de-la-comunicaci%C3%B3n-v2v-y-v2i-2023-08-5051384309-futurism

[15] Cuando el coche está conectado con la infraestructura: así funciona la comunicación V2I |. Accedido: 28 de marzo de 2024. Disponible en: https://www.race.es/v2i-coche-conectado-con-la-infraestructura

[16] What is ADAS (Advanced Driver Assistance Systems). Accedido: 28 de marzo de 2024.Disponible en: https://dewesoft.com/blog/what-is-adas

[17] Los ADAS obligatorios desde julio 2022, uno a uno. Accedido: 29 de marzo de 2024. Disponible en: https://revista.dgt.es/es/motor/tecnologia-seguridad/2021/0518-Landing-ADAS.shtml

[18] Validation of an Assistance System for Merging Maneuvers in Highways in Real Driving Conditions. Accedido: 9 de mayo de 2024. Disponible en: https://scholar.google.com/citations?view_op=view_citation&hl=es&user=eJMKna0AAAAJ&citation_for_view=eJMKna0AAAAJ:hFOr9nPyWt4C

[19] Qué es el ESP de un coche. Accedido: 2 de abril de 2024. Disponible en: https://www.race.es/que-es-el-control-de-estabilidad-esp-de-un-coche-y-como-actua

[20] PlatformIO. Accedido: 8 de abril de 2024. Disponible en: https://www.linkedin.com/pulse/platformio-introducci%C3%B3n-diego-hinojosa-c%C3%B3rdova/?originalSubdomain=es

[21] KICAD: el software estrella para el diseño de PCB. Accedido: 14 de abril de 2024. Disponible en: https://alfaiot.com/iot/kicad/

[22] Resistencias Pull-Up y Pull-Down. Accedido: 14 de abril de 2024. Disponible en: https://naylampmechatronics.com/blog/39_resistencias-pull-up-y-pull-down.html

[23] Módulo Buzzer Activo. Accedido: 14 de abril de 2024. Disponible en: https://avelectronics.cc/producto/buzzer-activo/

[24] Buzzer pasivo. Accedido: 14 de abril de 2024. Disponible en: https://www.taloselectronics.com/blogs/tutoriales/buzzer-pasivo

[25] J. Arquero Gallego, «Diseño y construcción de módulo IOT para la monitorización de la calidad del agua de riego», 2022.

[26] PCB FR4: La guía del FR-4 para sus circuitos impresos. Accedido: 18 de abril de 2024. Disponible en: https://www.proto-electronics.com/es/blog/pcb-fr4-la-guia-del-fr-4-para-sus-circuitos-impresos

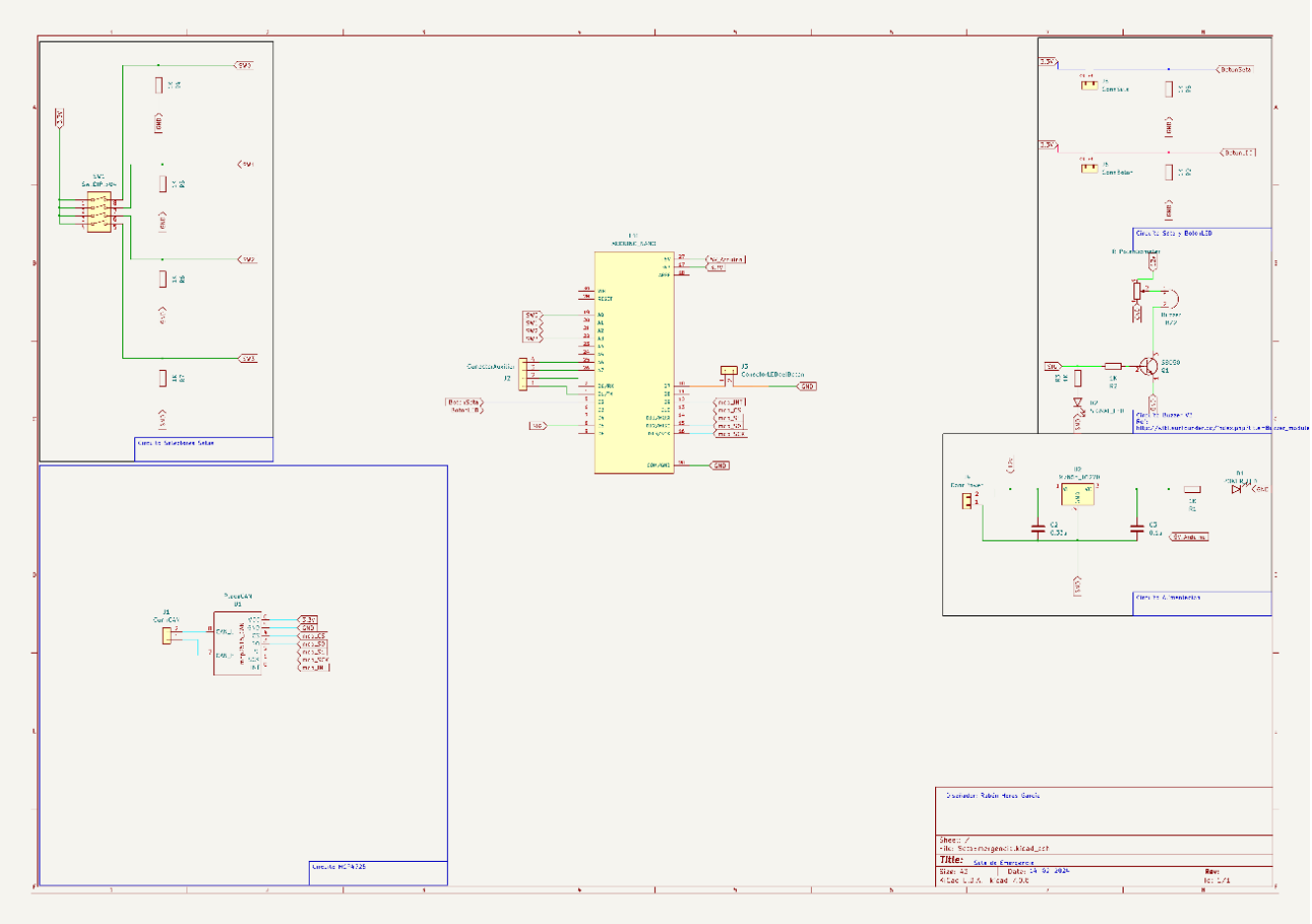
[27] SysML: Modelando Sistemas y Sistemas de Sistemas. Accedido: 25 de abril de 2024. Disponible en: https://www.abiztar.com.mx/articulos/sysml-modelando-sistemas-y-sistemas-de-sistemas.html

[28] Librería Tone. Accedido: 22 de abril de 2024. Disponible en: https://arduwiki.perut.org/index.php/Tone

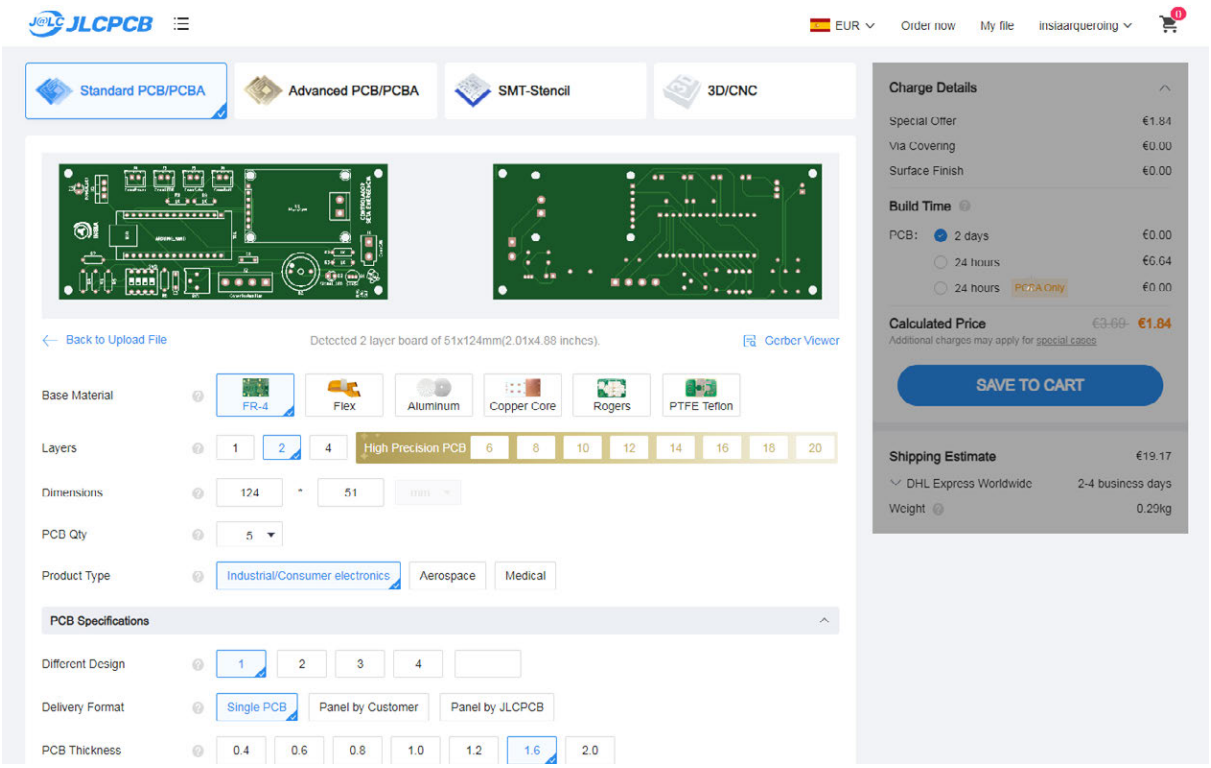
8. Anexos

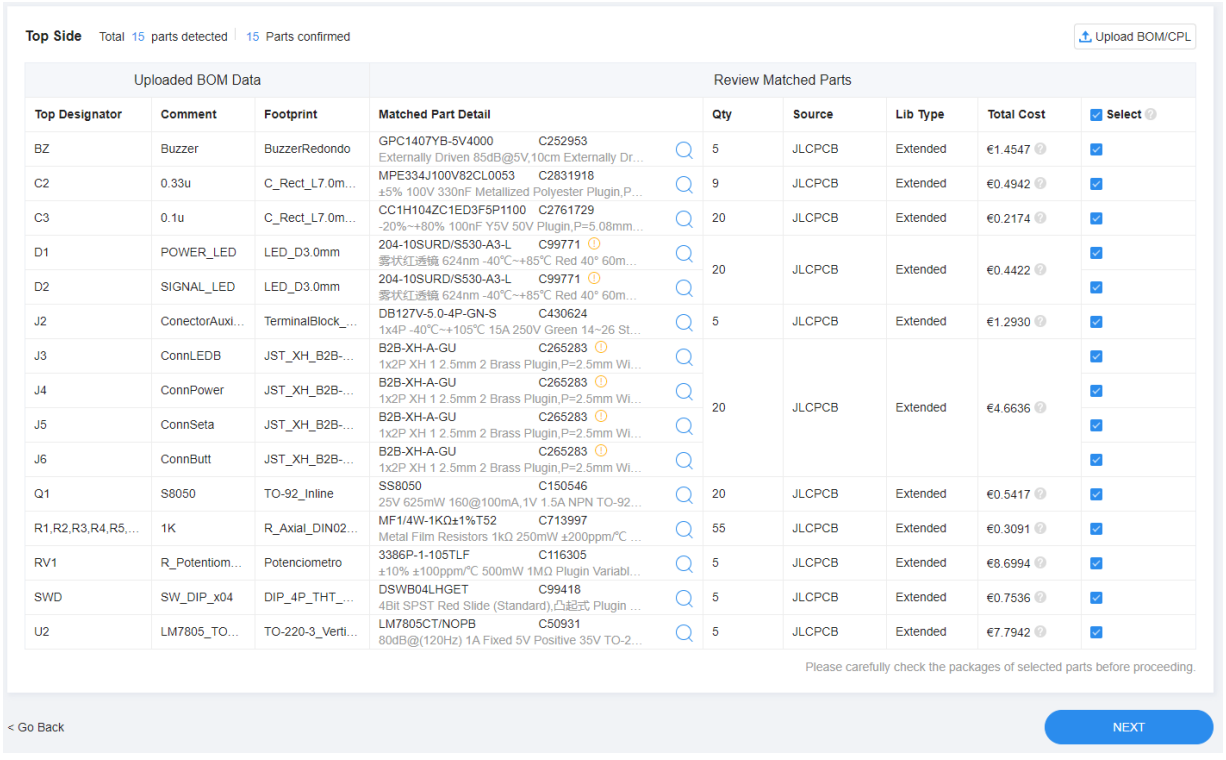
En este apartado se adjuntarán los esquemas y s que han sido importantes a lo largo del proyecto y que serán de gran ayuda para una mayor comprensión de este.

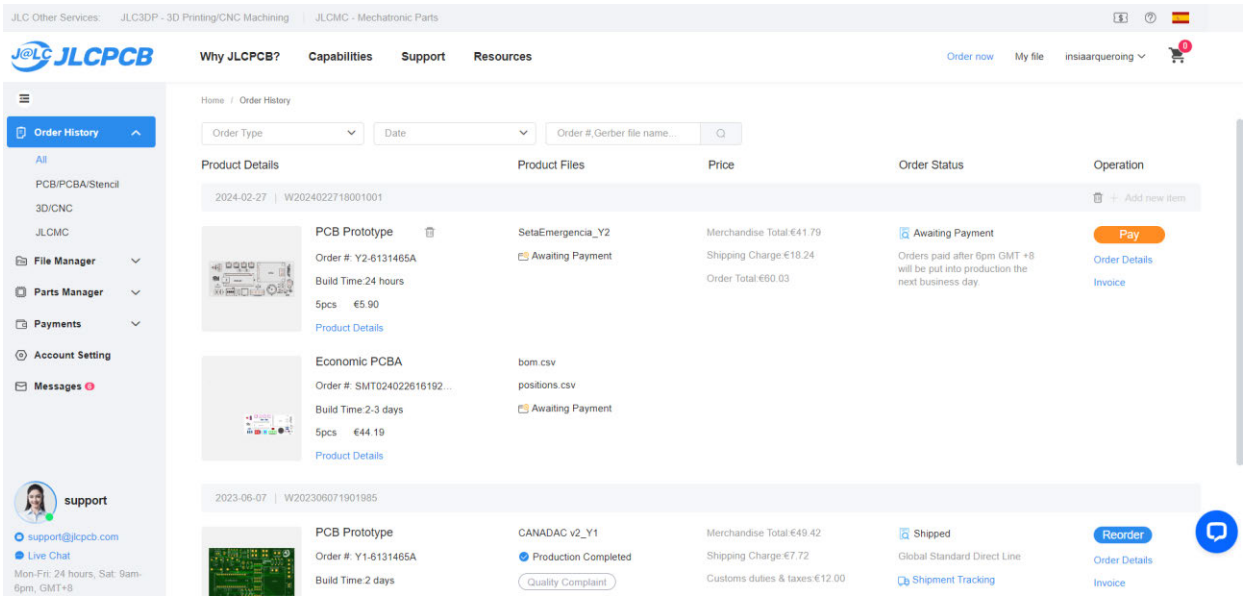
Anexo I – Esquemático Kicad



Anexo II – Compra Seta de Emergencia







Anexo III – Código Implementado Completo

```
// /*En este código se probará la Seta de emergencia para un vehículo. Consiste en que si se pulsa por primera vez la seta, el sistema entrará en modo PULSADO
//  * teniendo subir la seta para que el sistema quede en modo DESARMADO. Cuando sea pulsado el boton LED el sistema queda ARMADO y hasta que no se pulse otra vez la seta
//  * el sistema no volverá al modo PULSADO.
//  *
//  * A su vez, el boton LED se tendrá que iluminar cuando el sistema está ARMADO, apagándose cuando el sistema está DESARMADO.
//  * El sistema también es capaz de enviar mensajes CAN según el estado del sistema.
//  */



#include <Arduino.h>
#include <SPI.h>
#include <mcp2515.h>


MCP2515 mcp2515(10); // Set CS to pin 10


const int pinSetaEmergencia = 2;
const int pinBotonNormal = 3;
const int LEDPin = 7;
const int buzzer = 5;
uint8_t mensajeEnviado = MCP2515::ERROR::ERROR_NOMSG;

struct can_frame stmp;

enum EstadoSistema {
  DESARMADO,
  PULSADO,
  ARMADO
};

int getDIPvalue();



int numeroSeta;

unsigned long lastTime = 0;  // Almacena la última vez que se ejecutó getDIPvalue

volatile bool emergenciaPulsada = false;
volatile bool botonNormalPulsado = false;

unsigned long lastBuzzerTime = 0;
bool buzzerActive = false;


EstadoSistema estado = DESARMADO;

//-------------------FUNCIONES AUXILIARES-------------------
void interrupcionEmergencia() {
  emergenciaPulsada = true;
}

void interrupcionBotonNormal() {
  botonNormalPulsado = true;
}

int getDIPvalue(){
  bool d9, d4, d5, d6;
  int resul=0;
  d9=digitalRead(A0);
  d4=digitalRead(A1);
  d5=digitalRead(A2);
  d6=digitalRead(A3);
  if(d9 == true)resul+=1;
  if(d4 == true)resul+=2;
  if(d5 == true)resul+=4;
  if(d6 == true)resul+=8;
  return resul;
}
```

```
//------------------FIN FUNCIONES AUXILIARES------------------


void setup() {
  Serial.begin (9600);

  mcp2515.reset();
  mcp2515.setBitrate(CAN_500KBPS, MCP_8MHZ);   // set baudrate to 500kbps
  mcp2515.setConfigMode();     // set config mode
  mcp2515.setNormalMode();     // set normal mode to send and receive data.

  if(mcp2515.readMessage(&stmp) == MCP2515::ERROR_OK)
  {
    Serial.println("CAN BUS init NO OK");
  }else
  {
    Serial.println("CAN BUS init OK");
  }
  pinMode(pinSetaEmergencia, INPUT_PULLUP);
  pinMode(pinBotonNormal, INPUT_PULLUP);
  pinMode(LEDPin, OUTPUT);
  pinMode(buzzer, OUTPUT);


  attachInterrupt(digitalPinToInterrupt(pinSetaEmergencia), interrupcionEmergencia, FALLING);
  attachInterrupt(digitalPinToInterrupt(pinBotonNormal), interrupcionBotonNormal, FALLING);

  numeroSeta = getDIPvalue();

  //emergenciaPulsada = true;
}



void loop() {
  int estadoSetaEmergencia = digitalRead(pinSetaEmergencia);
  int estadobotonNormal = digitalRead(pinBotonNormal);

  unsigned long currentTime = millis();  // Obtiene el tiempo actual

  // Comprueba si ha pasado un segundo desde la última vez que se ejecutó getDIPvalue y si el estado está en DESARMADO, solo si esas dos condiciones son ciertas entonces se puede elegir otro número de seta
  if (currentTime - lastTime >= 1000 && estado == DESARMADO) {
    numeroSeta = getDIPvalue();  // Ejecuta getDIPvalue
   lastTime = currentTime;  // Actualiza la última vez que se ejecutó getDIPvalue
  }
      stmp.can_id=0x602;
      stmp.can_dlc=8;
      stmp.data[0]=0x00;//Specifier
      stmp.data[1]=0x00;//indice
      stmp.data[2]=0x00;//indice
      stmp.data[3]=0x00;//Subidnice
      stmp.data[4]=0x00;//datos
      stmp.data[5]=0x00;//datos
      stmp.data[6]=numeroSeta;//datos número de seta
      stmp.data[7]=0x00;//datos estado de seta


    //MIRAR ESTE WHILE
    switch (estado) {

      case PULSADO:
        stmp.can_id=0x602;
        stmp.can_dlc=8;
        stmp.data[0]=0x00; //Specifier
        stmp.data[1]=0x00; //indice
        stmp.data[2]=0x00; //indice
        stmp.data[3]=0x00; //Subidnice
        stmp.data[4]=0x00; //datos
        stmp.data[5]=0x00;
        stmp.data[6]=numeroSeta;
        stmp.data[7]=0x03;

        delay(50);
       //ESTO ES PARA CONTROLAR EL BUZZER QUE SE ENCIENDA Y SE APAGUE CADA SEGUNDO

       if(millis() - lastBuzzerTime >= 1000){
        if(buzzerActive){
          digitalWrite(buzzer, LOW); //Apagar el buzzer si está activo
          digitalWrite(LEDPin, LOW); //Apagar el LED si está activo
        }else{
          digitalWrite(buzzer, HIGH); //Encender el buzzer si está apagado
          digitalWrite(LEDPin, HIGH); //Encender el LED si está apagado
        }
        buzzerActive = !buzzerActive; //Cambiar el estado del buzzer
        lastBuzzerTime = millis(); //Actualizar el tiempo de la última vez que se cambió el estado del buzzer
       }
```

```
158
159           //-------------------------------------------------------------------------
160           if (estadoSetaEmergencia == HIGH) {
161             estado = DESARMADO;
162             delay(50);
163             digitalWrite(buzzer, LOW); //AQUI APAGO EL BUZZER PARA QUE DEJE SE SONAR
164             //mcp2515.sendMessage(&stmp);
165             //Serial.println(stmp.data[0]);
166             Serial.println("Modo SETA 1 DESARMADO ");
167             digitalWrite(LEDPin, LOW);
168             delay(100);
169           }
170           if(mcp2515.readMessage(&stmp) == MCP2515::ERROR_OK) //Si recibo un mensaje CAN que tiene identificador 0x20 significa que me están pidiendo información de mi estado
171           {
172             if (stmp.data[0] == 0x20)
173             {
174             stmp.can_id=0x602;
175             stmp.can_dlc=8;
176             stmp.data[0]=0x40; //Specifier
177             stmp.data[1]=0x00; //indice
178             stmp.data[2]=0x00; //indice
179             stmp.data[3]=0x00; //Subidnice
180             stmp.data[4]=0x00; //datos
181             stmp.data[5]=0x00;
182             stmp.data[6]=numeroSeta;
183             stmp.data[7]=0x03;
184
185             do
186             {
187               mensajeEnviado = mcp2515.sendMessage(&stmp);
188             } while (mensajeEnviado != MCP2515::ERROR::ERROR_OK);
189
190             }
191           }
192           do
193             {
194               mensajeEnviado = mcp2515.sendMessage(&stmp);
195             } while (mensajeEnviado != MCP2515::ERROR::ERROR_OK);
196           break;
197
198         case DESARMADO:
199           digitalWrite(LEDPin, LOW);
200           stmp.can_id=0x602;
201           stmp.can_dlc=8;
202           stmp.data[0]=0x00; //Specifier
203           stmp.data[1]=0x00; //indice
204           stmp.data[2]=0x00; //indice
205           stmp.data[3]=0x00; //Subidnice
206           stmp.data[4]=0x00; //datos
207           stmp.data[5]=0x00;
208           stmp.data[6]=numeroSeta;
209           stmp.data[7]=0x01;
210           delay(50);
211
212           if(estadobotonNormal == LOW){
213             do
214             {
215               mensajeEnviado = mcp2515.sendMessage(&stmp);
216             } while (mensajeEnviado != MCP2515::ERROR::ERROR_OK);;
217             Serial.println("Envio Mensaje DESARMADO");
218             unsigned long startMillis = millis();  // Guarda el tiempo actual
219
220             //ESTO ES PARA QUE SI PULSO LA SETA DE EMERGENCIA NO SE CUENTEN ESOS 5 SEGUNDOS Y SE SALGA DEL IF
221             while(estadobotonNormal== LOW){
222               if(millis() - startMillis >= 1000){
223                 break;
224             }
225             if(mcp2515.readMessage(&stmp) == MCP2515::ERROR_OK) //ESTO ES PARA QUE SI RECIBO UN MENSAJE CAN DE IDENTIFICADOR 0x20 Y DATO 0x20, SALGA DEL WHILE
226             {
227               if(stmp.data[0] == 0x20)
228               {
229                 stmp.data[0]=0x40;
230                 stmp.data[6]=numeroSeta;
231                 stmp.data[7]=0x01;
232                 do
233                 {
234                   mensajeEnviado = mcp2515.sendMessage(&stmp);
235                 } while (mensajeEnviado != MCP2515::ERROR::ERROR_OK);
236                     break;
237               }
238             }
239             estadobotonNormal = digitalRead(pinBotonNormal);
240           }
```

```
    }else{
    if (botonNormalPulsado == true) {
      stmp.can_id=0x602;
      stmp.can_dlc=8;
      stmp.data[0]=0x00; //Specifier
      stmp.data[1]=0x00; //indice
      stmp.data[2]=0x00; //indice
      stmp.data[3]=0x00; //Subidnice
      stmp.data[4]=0x00; //datos
      stmp.data[5]=0x00;
      stmp.data[6]=numeroSeta;
      stmp.data[7]=0x02;
      estado = ARMADO;
      do
      {
        mensajeEnviado = mcp2515.sendMessage(&stmp);
      } while (mensajeEnviado != MCP2515::ERROR::ERROR_OK);
      Serial.println("Sistema  SETA 1 ARMADO");
    }

    if(mcp2515.readMessage(&stmp) == MCP2515::ERROR_OK) //Si recivo un mensaje CAN que tiene identificador 0x20 significa que me están pidiendo información de mi estado
    {
      if (stmp.data[0] == 0x20)
      {
      stmp.can_id=0x602;
      stmp.can_dlc=8;
      stmp.data[0]=0x40; //Specifier
      stmp.data[1]=0x00; //indice
      stmp.data[2]=0x00; //indice
      stmp.data[3]=0x00; //Subidnice
      stmp.data[4]=0x00; //datos
      stmp.data[5]=0x00;
      stmp.data[6]=numeroSeta;
      stmp.data[7]=0x02;

      do
      {
        mensajeEnviado = mcp2515.sendMessage(&stmp);
      } while (mensajeEnviado != MCP2515::ERROR::ERROR_OK);
      }
    }
    }
    break:

  case ARMADO:
    digitalWrite(LEDPin, HIGH);

    stmp.can_id=0x602;
    stmp.can_dlc=8;
    stmp.data[0]=0x00; //Specifier
    stmp.data[1]=0x00; //indice
    stmp.data[2]=0x00; //indice
    stmp.data[3]=0x00; //Subidnice
    stmp.data[4]=0x00; //datos
    stmp.data[5]=0x00;
    stmp.data[6]=numeroSeta;
    stmp.data[7]=0x02;


    if(estadoSetaEmergencia == HIGH){
      do
      {
        mensajeEnviado = mcp2515.sendMessage(&stmp);
      } while (mensajeEnviado != MCP2515::ERROR::ERROR_OK);
      Serial.println("Envío Mensaje");
      unsigned long startMillis = millis();  // Guarda el tiempo actual

      //ESTO ES PARA QUE SI PULSO LA SETA DE EMERGENCIA NO SE CUENTEN ESOS 5 SEGUNDOS Y SE SALGA DEL IF
      while(estadoSetaEmergencia == HIGH){
        if(millis() - startMillis >= 5000){
          break;
        }

      if(mcp2515.readMessage(&stmp) == MCP2515::ERROR_OK) //ESTO ES PARA QUE SI RECIBO UN MENSAJE CAN DE IDENTIFICADOR 0x20 Y DATO 0x20, SALGA DEL WHILE
      {
        if(stmp.data[0] == 0x20)
        {
          stmp.can_id=0x602;
          stmp.can_dlc=8;
          stmp.data[0]=0x40; //Specifier
          stmp.data[1]=0x00; //indice
          stmp.data[2]=0x00; //indice
          stmp.data[3]=0x00; //Subidnice
          stmp.data[4]=0x00; //datos
          stmp.data[5]=0x00;
          stmp.data[6]=numeroSeta;
          stmp.data[7]=0x02;

          do
          {
            mensajeEnviado = mcp2515.sendMessage(&stmp);
          } while (mensajeEnviado != MCP2515::ERROR::ERROR_OK);
              break;
        }
      }

      estadoSetaEmergencia = digitalRead(pinSetaEmergencia);
    }
```

```
339             }else{
340             if (emergenciaPulsada == true) {
341                 emergenciaPulsada = false;
342                 estado = PULSADO;
343                 digitalWrite(LEDPin, LOW);
344             }
345             if(mcp2515.readMessage(&stmp) == MCP2515::ERROR_OK) //Si recivo un mensaje CAN que tiene identificador 0x20 significa que me están pidiendo información de mi estado
346             {
347               if (stmp.data[0] == 0x20)
348               {
349               stmp.can_id=0x602;
350               stmp.can_dlc=8;
351               stmp.data[0]=0x40; //Specifier
352               stmp.data[1]=0x00; //indice
353               stmp.data[2]=0x00; //indice
354               stmp.data[3]=0x00; //Subidnice
355               stmp.data[4]=0x00; //datos
356               stmp.data[5]=0x00;
357               stmp.data[6]=numeroSeta;
358               stmp.data[7]=0x02;
359
360               do
361               {
362                 mensajeEnviado = mcp2515.sendMessage(&stmp);
363               } while (mensajeEnviado != MCP2515::ERROR::ERROR_OK);
364               }
365             }
366             }
367             break;
368         }
369 }
```

Anexo IV – Pruebas Unitarias

- Boton LED

```
const int pinBotonNormal = 3;


void setup() {
  Serial.begin (9600);
  pinMode(pinBotonNormal, INPUT_PULLUP);


}



void loop() {
  int estadoSetaEmergencia = digitalRead(pinBotonNormal);

  if (estadoSetaEmergencia == LOW) {
    Serial.println("Boton no pulsado");
  }
  else {
    Serial.println("Boton pulsado");
  }
}
```

- Buzzer

```
const int buzzer = 5;
unsigned long lastBuzzerTime = 0;
bool buzzerActive = false;

void setup() {
  Serial.begin (9600);
   pinMode(buzzer, OUTPUT);


}


void loop() {
  if(millis() - lastBuzzerTime >= 1000){
            if(buzzerActive){
              digitalWrite(buzzer, LOW); //Apagar el buzzer si está activo
              digitalWrite(LEDPin, LOW); //Apagar el LED si está activo
            }else{
              digitalWrite(buzzer, HIGH); //Encender el buzzer si está apagado
              digitalWrite(LEDPin, HIGH); //Encender el LED si está apagado
            }
            buzzerActive = !buzzerActive; //Cambiar el estado del buzzer
            lastBuzzerTime = millis(); //Actualizar el tiempo de la última vez que se cambió el estado del buzzer
          }
}
```

- MCP2515

```
MCP2515 mcp2515(10); // Set CS to pin 10
struct can_frame stmp;
void setup() {
  Serial.begin (9600);

  mcp2515.reset();
  mcp2515.setBitrate(CAN_500KBPS, MCP_8MHZ);   // set baudrate to 500kbps
  mcp2515.setConfigMode();     // set config mode
  mcp2515.setNormalMode();     // set normal mode to send and receive data.

  if(mcp2515.readMessage(&stmp) == MCP2515::ERROR_OK)
  {
     Serial.print("CAN BUS init NO OK");
  }else
  {
     Serial.print("CAN BUS init OK");
  }
}

//ESTE CODIGO ENVÍA UN MENSAJE CAN CADA 500ms

void loop() {

       stmp.can_id=0x602;
       stmp.can_dlc=8;
       stmp.data[0]=0x00;//Specifier
       stmp.data[1]=0x00;//indice
       stmp.data[2]=0x00;//indice
       stmp.data[3]=0x00;//Subidnice
       stmp.data[4]=0x00;//datos
       stmp.data[5]=0x00;//datos
       stmp.data[6]=0x01;//datos
       stmp.data[7]=0x00;//datos

          mcp2515.sendMessage(&stmp);
          Serial.println("ENTRO EN EL WHILE");
          delay(500);

}
```

- DipSwitch

```
const int LEDPin = 7;

bool d9, d4, d5, d6;
int resul = 0;

void setup() {
  Serial.begin (9600);
  pinMode (d9, INPUT);
  pinMode (d4, INPUT);
  pinMode (d5, INPUT);
  pinMode (d6, INPUT);
}



void loop() {
  d9 = analogRead(0);
  d4 = analogRead(1);
  d5 = analogRead(2);
  d6 = analogRead(3);

  if (d9 == HIGH) {
    resul = 1;
  }
  if (d4 == HIGH) {
    resul = 2;
  }
  if (d5 == HIGH) {
    resul = 3;
  }
  if (d6 == HIGH) {
    resul = 4;
  }
  Serial.println(resul);
  delay(1000);
}
```

- LED

```
const int LEDPin = 7;



void setup() {
  Serial.begin (9600);

  pinMode(LEDPin, OUTPUT);

}



void loop() {
  digitalWrite(LEDPin, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LEDPin, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);
}
```

- Switch Seta

```
#include <Arduino.h>
#include <SPI.h>
#include <mcp2515.h>


const int pinSetaEmergencia = 2;


void setup() {
  Serial.begin (9600);
  pinMode(pinSetaEmergencia, INPUT_PULLUP);


}



void loop() {
  int estadoSetaEmergencia = digitalRead(pinSetaEmergencia);

  if (estadoSetaEmergencia == LOW) {
    Serial.println("Seta pulsada");
  }
  else {
    Serial.println("Seta no pulsada");
  }
}
```

Para más información sobre el código programado y los archivos para la impresión de la PCB, acceder al siguiente enlace de GitHub[5].

[5] https://github.com/SIMCA-USI/Seta_Emergencia/tree/Version8